Performance von Webseiten verbessern S. 30

Objektorientierung in JavaScript S. 36

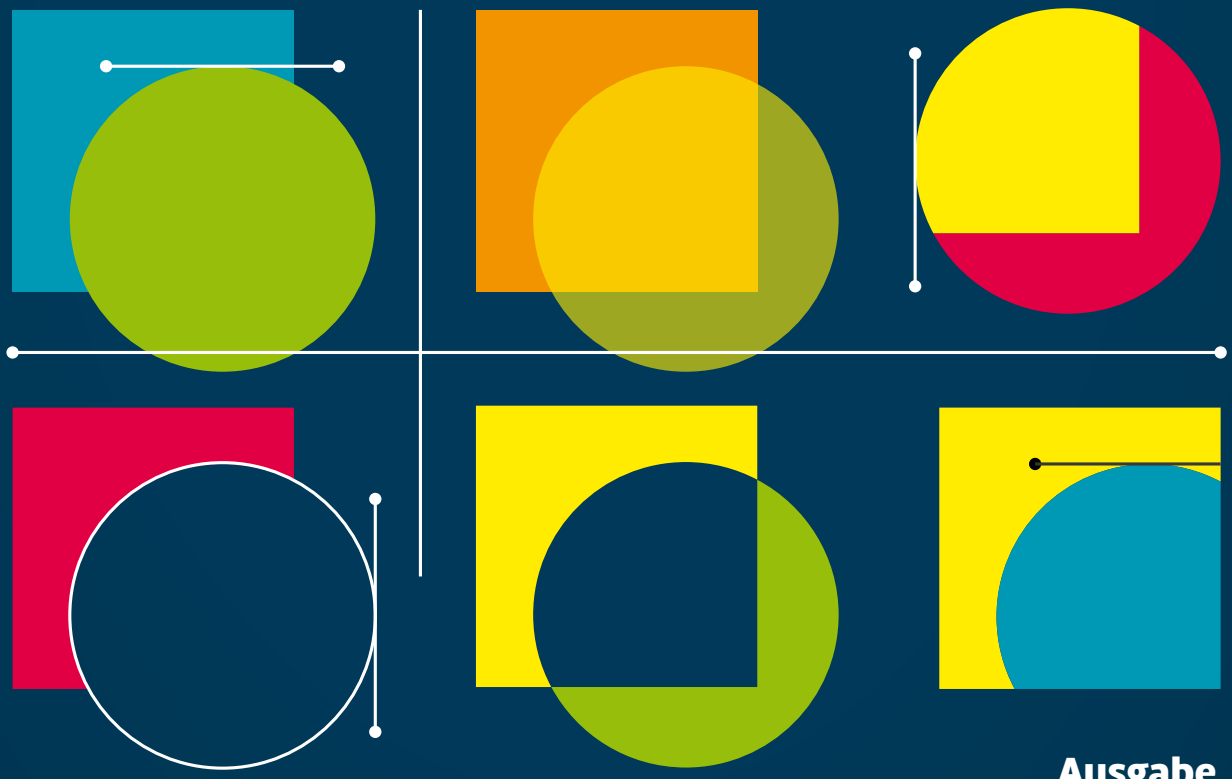
Windows 10 für Entwickler S. 126

web & mobile DEVELOPER

webundmobile.de

Webdesign mit HTML5

So integrieren Sie Grafiken mit <canvas> und JavaScript



Auf der Heft-CD

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

INFO-Programm gemäß § 14 JuSchG

Ausgabe 7/16

Deutschland 14,95 EUR

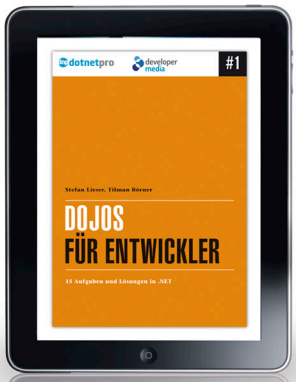
CH: 29,90 CHF
A, B, NL, L: 16,45 EUR

4 198255 014951 07

developer media Shop

Upgrades für Ihr Entwickler-Know-How

Ab sofort in unseren Shops erhältlich!



https://shop.dotnetpro.de

https://shop.webundmobile.de

Webdesign mit HTML5

Das HTML5-Element <canvas> hat eine Vielfalt an innovativen Lösungen auf der Basis von JavaScript hervorgebracht.

Das moderne Web möchte den Benutzer nicht nur mit üppiger Grafik und bewegungsreichen Animationen verwöhnen, sondern auch durch ausgefallene Interaktivität engagieren. Die zwei führenden Technologien, HTML5 mit dem <canvas>-Element in Kombination mit CSS3 und geeigneten JavaScript-Bibliotheken, SVG und WebGL bieten eine enorme Vielfalt an technischen Möglichkeiten. Noch nie zuvor standen Webentwicklern so viele verschiedene Technologien zur Verfügung, um diese Ziele auch wirklich zu erreichen. Unsere Autoren Anna Kobylinska und Filipe Martins haben in einem Artikel ab Seite 14 einige dieser Technologien unter die Lupe genommen.

»Noch nie zuvor standen Webentwicklern so viele verschiedene Technologien zur Verfügung.«

JavaScript ist keine klassenbasierte Programmiersprache. Daran ändert auch die in ES6/ES2015 eingeführte Klassensyntax nichts, die oberflächlich Ähnlichkeiten etwa zur von Java verwendeten Syntax aufweist und daher für Spracheinsteiger den Schluss aufkommen lassen könnte, JavaScript sei klassenbasiert. Doch JavaScript ist eine sogenannte objektbasierte Programmiersprache: Objektinstanzen werden hier nicht auf Basis von Klassen, sondern auf Basis von anderen Objekten (den Prototypen) erstellt. Ein Artikel von Philip Ackermann ab Seite 36 widmet sich ausführlich diesem Thema.

Die Klasse UITableView gehört wohl zusammen mit ihrem View-Controller UITableViewController zu den am häufigsten verwendeten Klassen in der iOS-Entwicklung. Dabei ist der Name eigentlich ein wenig irreführend: Eine UITableView listet zwar einzelne Zellen in einer vertikalen Reihe auf, das war es aber auch schon. Spalten – wie man sie typischerweise von Tabellen kennt – sind mit einer UITableView von Haus aus nicht realisierbar. Wer also eine Tabelle mit Spalten in eigenen iOS-Apps benötigt, muss selbst Hand anlegen. Wie, das zeigt Thomas Sillmann in einem Artikel ab Seite 78.

Ihr Max Bold
chefredakteur@maxbold.de

Dr. Florence Maurice

untersucht die Rendering-Performance von Webseiten und Web-Apps (S. 30)

Frank Simon

zeigt, wie man auf eine NoSQL-Datenbank umsteigt (S. 92)

Johann Baumeister

stellt OpenStack und seine Dienste vor (S. 108)

INHALT

Aktuell

Open-Source-CMS
Pagekit 1.0 ist verfügbar 6

Feature

Canvas-Frameworks
Das HTML5-Element Canvas lässt sich mit einem geeigneten Framework viel einfacher nutzen 14

HTML, CSS & JavaScript

Rendering-Performance von Webseiten und Web-Apps
Grundlegende Konzepte, Techniken und Tools, die bei der Überprüfung der Rendering-Performance helfen 30

Objektorientierung in JavaScript
In JavaScript sieht Objektorientierung anders aus als in Sprachen wie Java 36

Datenvisualisierung mit D3
D3 ist eine mächtige JavaScript-Bibliothek zur Visualisierung großer Datenmengen 46

Web Storage API
Über das Web Storage API können Webentwickler Daten lokal auf Clientseite speichern 52

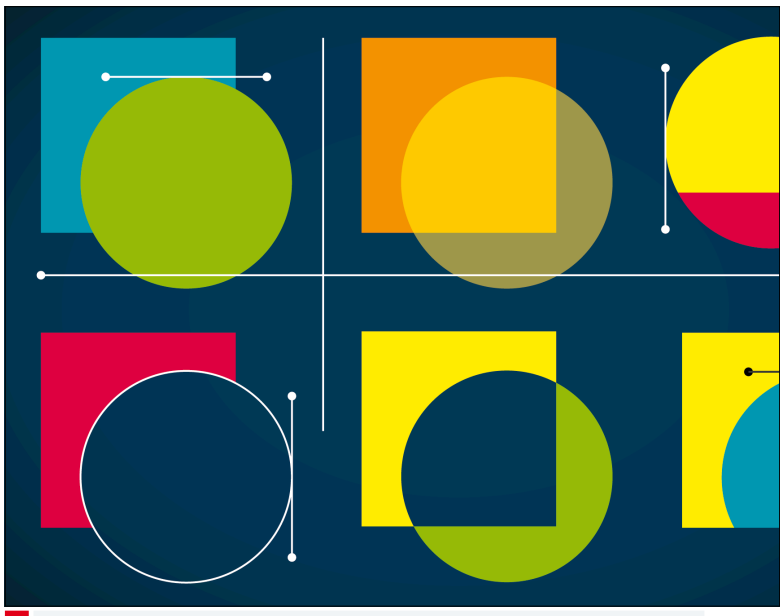
Mobile Development

Die wichtigsten Neuerungen und Änderungen von Swift 2.2
Mit Swift 2.2 legt Apple den Grundstein für die kommende Version 3 von Swift und lenkt die Sprache in die richtige Richtung 58

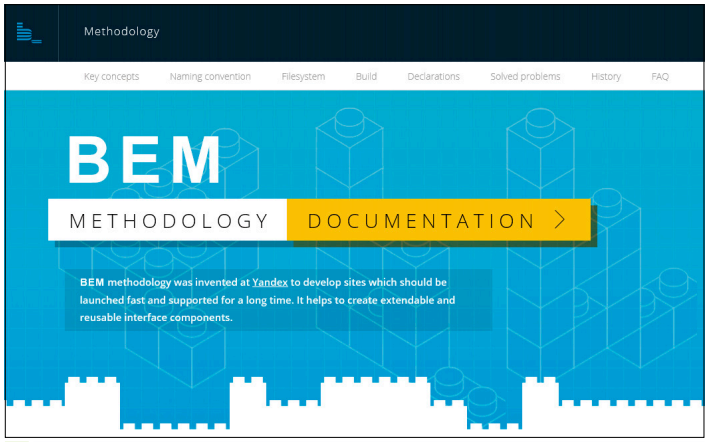
Apples 3D Touch im iPhone
Wie man Apples 3D Touch innerhalb einer App nutzen kann 64

Spieleprogrammierung unter OpenGL (Teil 3)
Wirklich interessant wird ein Game erst dann, wenn es auch eine Handlung aufweist 68

Erstellen einer eigenen Table-View mit Spalten
Eine eigene Table-View-Klasse entwickeln, die neben Zeilen auch über Spalten verfügt 78

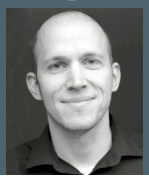


Wenn es um JavaScript-Bibliotheken und Grafik-APIs geht, haben Entwickler jetzt die Qual der Wahl S. 14



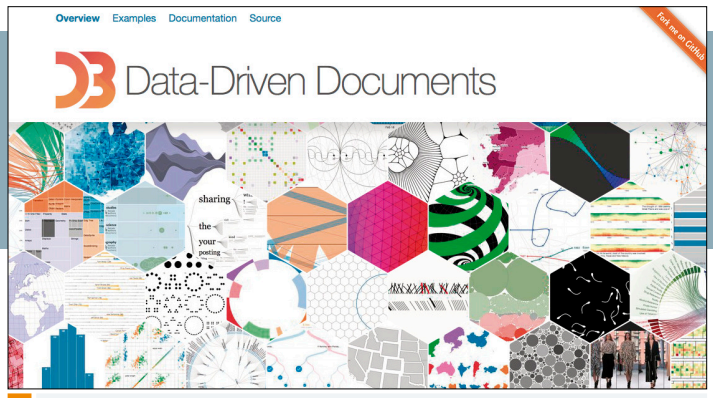
BEM (Block, Element, Modifier) ist eine vom Yandex-Team entwickelte Methode, um CSS-Selektoren zu benennen S. 30

Experten in dieser Ausgabe

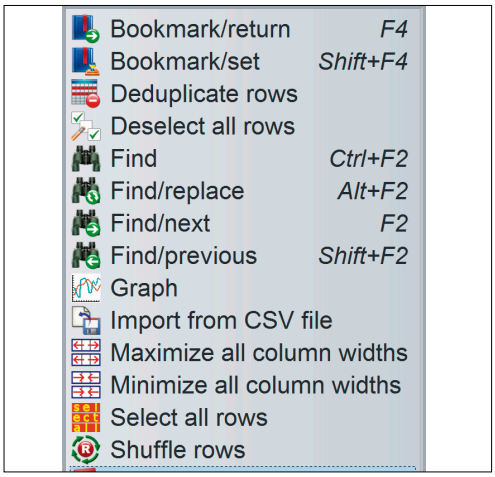


Philip Ackermann untersucht, wie es um die Objektorientierung in der Skriptsprache JavaScript steht. 36

Thomas Sillmann stellt die Neuerungen der Version 2.2 von Apples Programmiersprache Swift vor. 58



D3-Website: Die D3-Homepage ist für Entwickler eine Fundgrube in Sachen Datenvisualisierung S. 46



Umstieg auf Neo4j 2: Der CSV Editor Pro bietet mächtige Funktionen. Neben einem Filter zum Ausblenden von Teilen der CSV-Datei besitzt das Programm unter anderem einen Multi-Line-Edit-Modus S. 92

App- und Webentwicklung für die Generation 65+: Anders als für jeden 20-Jährigen sind Doppelklick und Wischgesten für ältere Mediennutzer keine Selbstverständlichkeiten S. 118

Jetzt abonnieren

Sichern Sie sich jetzt die web & mobile developer im Jahresabo und profitieren Sie von exklusiven Services und Angeboten für Abonnenten.
https://shop.webundmobile.de

Backend

Neo4j: Umstieg in die reale Welt der Graphen
Neo4j besitzt komfortable Import-Schnittstellen, um vorhandene Datenbestände in eine Graph-Datenbank zu überführen 92

Template-Entwicklung für Magento 2.0
Auch bei der Template-Entwicklung hat sich bei der Version 2.0 von Magento einiges getan 104

OpenStack: Ein Framework für die Cloud
OpenStack ist eine Plattform für den Aufbau und den Betrieb von virtualisierten Systemen und Clouds 108

Migration von Inhalten zu WordPress
Wie man den Umzug eines bestehenden Systems zum Content-Primus richtig macht 114

Beyond Dev

Grafik für Entwickler: Silver Surfer
Internet und Smartphone für die Generation 65+ – eine Herausforderung für App- und Webentwickler 118

Umgang mit ortsbezogenen Informationen
Ortsbezogene Informationen und darauf basierende Dienste werden nicht erst seit Google Maps immer wichtiger 122

Build 2016: Windows 10 für Entwickler
Das Kernthema von Microsofts Entwickler-Konferenz Build 2016 war Windows 10 aus der Sicht der Entwickler 126

Android, iOS oder Windows Mobile: Mehr Sicherheit für die Endgeräte
Welches mobile Betriebssystem sich am besten für den Unternehmenseinsatz eignet 130

Magento Imagine 2016
Das wichtigste Magento-Event des Jahres, die Imagine, fand vom 11. bis 13. April in Las Vegas statt 136

Standards

Editorial 3
Heft-CD 50
Online-Recht 138
Impressum 141
Arbeitsmarkt 142
Dienstleisterverzeichnis 145
Vorschau 146

NEWS & TRENDS

AKTUELLE NEWS AUS DER ENTWICKLERSZENE

E-Commerce 2016

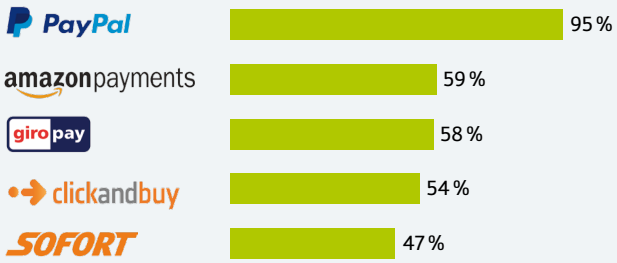
Payment in Zahlen

Das Thema Payment ist ein zentraler Erfolgsfaktor im deutschen Online-Handel. Schließlich entscheidet der richtige Angebots-Mix an Zahlungsmöglichkeiten oft über Kaufabbruch oder generierten Umsatz. Gleichzeitig muss eine Balance zwischen Händlerrisiko und -kosten auf der einen Seite und Konsumentenrisiko und -wünschen auf der anderen Seite hergestellt werden. Die Internet World Messe hat die wichtigsten aktuellen Kennzahlen rund um das Thema E-Commerce Payment in Deutschland gebündelt und in einer Infografik zusammengefasst. Die Grafik gibt Aufschluss über die beliebtesten Zahlungsarten im deutschen Online-Handel und zeigt, wie Händler mit Hilfe von Payment-Angeboten Kosten und Risiko steuern.

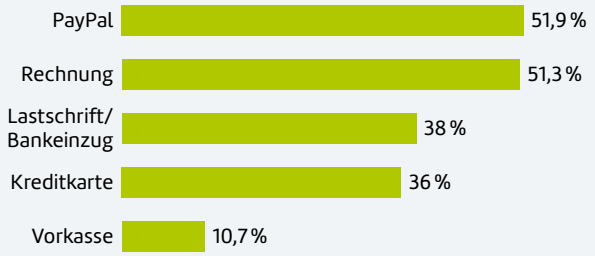
Wenig überraschend ist PayPal Deutschlands bekanntestes Online-Bezahlsystem. Durchaus überraschend dagegen ist, dass PayPal inzwischen sogar den Kauf per Rechnung in der Gunst der Online-Konsumenten knapp überholt hat. Vor allem männliche Online-Shopper bezahlen gerne darüber. In Sachen Bekanntheitsgrad rangieren auch Amazon Payments und Giropay unter den vordersten drei, und das, obwohl deutsche Händler mit der Integration des Amazon Zahlverfahrens sehr zögerlich sind.

www.internetworld-messe.de

Das sind Deutschlands 5 bekannteste Online-Bezahlsysteme:



Das sind die Top 5 der beliebtesten Zahlungswege im Web:



Die beliebtesten und bekanntesten Online-Bezahlsysteme in Deutschland

Quelle: Fittkau & Maaß

Open-Source-CMS

Erste Version von Pagekit verfügbar

Nach einer Entwicklungsphase von drei Jahren hat der Hamburger Theme- und Extension-Anbieter YOOtheme jetzt die erste Version seines Open-Source-CMS Pagekit vorgestellt.

Pagekit: Das neue Open-Source-CMS setzt auf bewährte Technologien

Pagekit ist ein modulares und schlankes CMS, das in PHP geschrieben ist und auf modernen Technologien wie Symfony-Komponenten und Composer basiert. Die Bedienoberfläche ist stark durch Googles Material Design beeinflusst und mit Vue.js gebaut. Als eines von wenigen CMS unterstützt Pagekit neben MySQL auch SQLite als zugrunde liegende Datenbank. Weitere Highlights von Pagekit: ein vorinstallierter Blog mit Kommentarfunktion und ein Media-Manager zur Verwaltung von Dateien, Bildern und anderen Assets für den Aufbau von Websites. Obendrein gibt es zum Start von Pagekit noch drei kostenlose Themes als Dreingabe. Mit TinyMCE steht ein populärer Wysiwyg-Editor als Add-on zur Verfügung.

Offenbar hat das CMS, das unter der MIT-Lizenz vertrieben wird, schon in der Entwicklungsphase zahlreiche Freunde gefunden, wie die Zahl von 20.000 registrierten Usern zeigt.

https://pagekit.com

Health Sciences Safety Suite

Oracle präsentiert Cloud-Version

Wenn es um die Markteinführung und den Verkauf von Medikamenten geht, sehen sich pharmazeutische Unternehmen und Organisationen mit immer strengeren Sicherheitsanforderungen konfrontiert. Dazu zählen das ordnungsgemäße Management quasi unbegrenzter Mengen an Sicherheitsdaten sowie eine korrekte und rasche Berichterstattung.

Aktuell dazu hat Oracle Health Sciences nun eine neue Cloud-Version der Oracle Health Sciences Safety Suite 8.0 auf den Markt gebracht. Mit dieser Lösung können Arzneimittelhersteller, Sponsoren klinischer Prüfungen, Auftragsforschungsinstitute und die Gesundheitsbehörden schnellere

Umfassende Cloud-Lösung macht Medikamente noch sicherer

Zahl des Monats

Laut einer Umfrage von »BRAVO« haben **96 %** der befragten Jugendlichen ein Smartphone. Für **91 %** ist WhatsApp die meistgenutzte Social-Media-App, gefolgt von YouTube **(56 %)** und Instagram **(52 %)**.

Quelle: BRAVO

und fundierte Entscheidungen treffen, wenn es um die Einhaltung von Sicherheitsstandards geht. Zudem können sie ihr Global Compliance Management verbessern und Risiken über den gesamten Lebenszyklus von Medikamenten ganzheitlich managen.

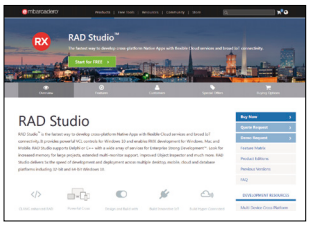
»In ihrem Bestreben, medikamentöse Therapien zu entwickeln, die sicher und gleichzeitig hochwirksam sind, brauchen Pharma-Unternehmen und Organisationen aus dem Bereich Life Sciences IT-Lösungen, mit denen sie Sicherheitsprozesse durchgängig automatisieren und so präziser machen können«, erläutert Steve Rosenberg, Vice President und General Manager bei Oracle Health Sciences. »Die Version 8.0 unserer Safety Suite kombiniert nun unser branchenführendes Know-how im Bereich Sicherheits- und Datenmanagement mit unserer Cloud-Plattform. Damit lassen sich Innovationen sehr viel schneller umsetzen.«

www.oracle.com

Embarcadero

Updates für RAD Studio, C++ Builder und Delphi

Embarcadero Technologies stellt ab sofort die Version 10.1, Codename Berlin, von RAD Studio, C++Builder und Delphi bereit. Die optimierten Versionen bringen verschiedene neue und verbesserte Funktionen, darunter spürbare Verbesserungen rund um Konnektivität für das Internet der Dinge (IoT) sowie Multi-Device-Vorschaufunktionen, mit denen sich die Produktivität der Entwickler steigern lässt.



RAD Studio: Neue Version mit verbesserten Funktionen

Version 10.1 baut die Möglichkeit für Programmierer aus, auf Basis einer einzigen Codequelle rasch Anwendungen für verschiedenste Endgeräte zu entwickeln.

Die Verbesserungen an FireUI sorgen für die Echtzeitvorschau der verschiedenen Geräteversionen. Zur neuen Version gehören auch Verbesserungen wie der GetIt Package Manager.

Er erleichtert das Erstellen von Komponenten und Erweiterungen für das große Feld von Embarcadero-Partnern. Wichtig ist dies vor allem im Hinblick auf Entwicklung und Support für eine breite Spanne an vernetzten IoT-Anwendungen.

»10.1 Berlin ist für Embarcadero ein wichtiger Meilenstein und untermauert unsere Konzentration auf Entwickler«, sagt Atanas Popov, General Manager bei Embarcadero. »Wir sind die erste Wahl für Entwickler, die sehr schnell wiederverwendbaren, plattformunabhängigen Code schreiben wollen. Zudem ist unsere Fähigkeit, für Enterprise-taugliche Anwendungen zu immens niedrigen Kosten zu sorgen, bis heute unerreicht.«

www.embarcadero.com

Progress

Umfrage unter .NET-Entwicklern

Die mehr als 1000 Befragten einer Umfrage von Progress sind der Meinung, dass Trends wie die plattformübergreifende Entwicklung und Open Source .NET zu einem deutlichen Aufschwung verhelfen.

Das Progress-Unternehmen Telerik hat dazu Entwickler nach ihren Ansichten zu Schlüsseltechnologien und Innovationen bei der Erstellung von mobilen, Desktop- und Webapplikationen

Progress hat die Ergebnisse seiner »2016 .NET Community Survey« veröffentlicht

befragt. Die Studie liefert Belege dafür, dass die Trends bei der plattformübergreifenden Entwicklung und bei Open Source eine spannende Zeit für .NET-Entwickler bringen.

Bei den bevorzugten Entwicklungsplattformen ergibt sich ein differenziertes Bild. Unter allen Befragten weltweit dominiert das Responsive Web (49 Prozent) als Zielplattform, gefolgt von Desktop-Browsern (45 Prozent) und Windows-Desktops (30 Prozent). In Deutschland dagegen kehrt sich die Reihenfolge um: Windows-Desktops (67 Prozent), Desktop-Browser (27 Prozent) und Responsive Web (18 Prozent)

Mehr als die Hälfte der Webentwickler erstellen mit Responsive-Web-Technologien Applikationen, die sowohl auf Desktops als auch auf mobilen Endgeräten laufen. Bei 38 Prozent der Desktop- und 78 Prozent der Webentwickler ist HTML die bevorzugte Sprache. Die große Mehrheit der Entwickler setzt JavaScript ein. Angular ist das beliebteste Core-Framework, gefolgt von Knockout.

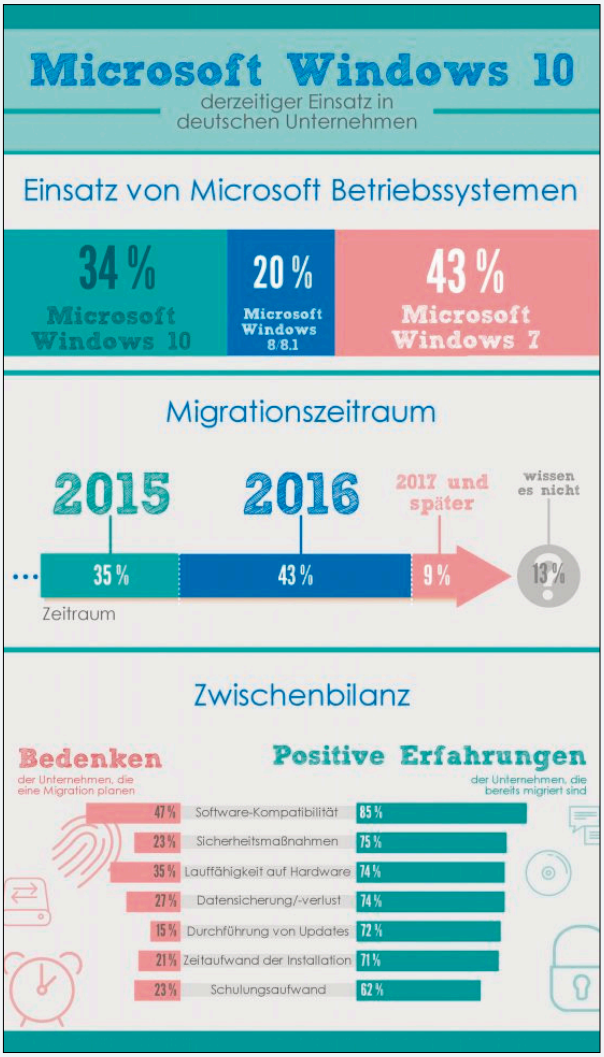
www.progress.de

Microsoft

Kostenloses Windows-10-Upgrade läuft aus

Am 29. Juli läuft das Angebot von Microsoft zum kostenlosen Upgrade auf Windows 10 für qualifizierte Rechner mit Windows 7 und Windows 8.1 aus. Eine aktuelle Research Note des Analystenhauses Techconsult zeigt, dass gerade kleine und mittelständische Unternehmen (KMU) in Deutschland die Chance zum Umstieg nutzen. So beträgt der Anteil von Windows-10-Geräten bereits 34 Prozent. Weitere 43 Prozent planen demnach eine Migration noch in diesem Jahr. Allein 31 Prozent der Firmen entschieden sich für Windows 10, da es gezielt auf Datenschutz ausgerichtet ist. 83 Prozent der Windows-10-User zeigen sich nach dem Umstieg zufrieden mit dem neuen Betriebssystem. Im Rahmen einer Breitenerhebung in 350 Unternehmen hat das Analystenhaus Techconsult außerdem im März dieses Jahres deutsche Firmen zum bisherigen Einsatz von Windows 10 befragt. Die Ergebnisse der Befragung haben die Analysten in einem Webcast sowie einer komprimierten Research Note festgehalten.

www.techconsult.de/blog



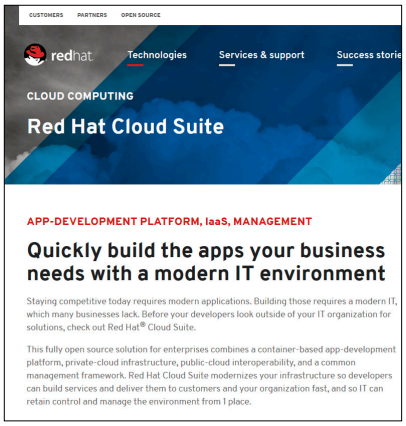
Windows 10-Upgrade: 34 Prozent der KMUs sind bereits umgestiegen

Red Hat

OpenStack Platform 8 und Red Hat Cloud Suite verfügbar

Die neuen Hybrid-Cloud-Lösungen von Red Hat unterstützen Linux-Container, integrieren Red Hat CloudForms sowie Red Hat Ceph Storage und ermöglichen so eine vereinfachte hybride Verwaltung sowie eine höhere Sicherheit und Performance.

Red Hat liefert ab sofort die Red Hat OpenStack Platform 8 sowie die Red Hat Cloud Suite aus und baut damit eine Brücke zwischen Entwicklung und Betrieb umfangreicher Cloud-Umgebungen.



Red Hat bietet eine Brücke zwischen Entwicklung und Betrieb umfangreicher Cloud-Umgebungen

In Form der Red Hat Cloud Suite bietet Red Hat seinen Kunden einen voll integrierten Hybrid Cloud Stack, bestehend aus der Container-Applikations-Plattform OpenShift by Red Hat, der massiv und unkompliziert skalierbaren Infrastruktur Red Hat OpenStack Platform 8 und den einheitlichen Verwaltungs-Tools von Red Hat CloudForms.

Red Hat OpenStack Platform 8 ist die neueste Version der hochskalierbaren IaaS-Plattform des Unternehmens, die auf dem OpenStack-Community-Release Liberty basiert und das Rückgrat von Red Hats Hybrid-Cloud-Lösungen bildet. Red Hat OpenStack Platform 8 kombiniert das bewährte Fundament von Red Hat Enterprise Linux mit der OpenStack-Technologie von Red Hat und bildet daraus eine für den Unternehmenseinsatz geeignete Cloud-Plattform, die sich zum Gold-Standard für umfangreiche Produktionsumgebungen entwickelt hat.

Hunderte von Lösungen befinden sich weltweit bereits im Einsatz und noch mehr Proof-of-Concepts sind angelaufen. In der IT- und TK-Branche unterstützt ein Ecosystem führender Hersteller die Red Hat OpenStack Platform und implementiert damit innovative Network-Functions-Virtualization- (NFV) und Software-defined-Networking-Lösungen (SDN).

Die Red Hat Cloud Suite kombiniert die Container-Applikations-Plattform OpenShift Enterprise mit der skalierbaren Cloud-Infrastruktur der Red Hat OpenStack Platform und dem Management-Framework Red Hat CloudForms. Unternehmen sind damit in der Lage, schneller neue Cloud-Services für Private Clouds und die Applikationsentwicklung bereitzustellen und vorhandene Workloads mit einer effizienteren Virtualisierung und einer höheren Performance zu optimieren. Application-Delivery-Teams können sich so – unter Beibehaltung der vollständigen Kontrolle und Transparenz – auf Entwicklung, Tests und Implementierung innovativer Cloud-basierter, containerisierter Microservice-Lösungen konzentrieren.

www.redhat.com/cloud-suite

DOMAINS | HOMEPAGES | CLOUD SERVER | WEBHOSTING

DAS SICHERSTE SICHER!

100% SICHERE TRANSAKTIONEN
SSL
INKLUSIVE

SSL-ZERTIFIKAT
HÖCHSTE SICHERHEIT
EXKLUSIV FÜR 1&1!

HOSTING? ABER SICHER!

1&1 bietet die höchsten Sicherheitsstandards für Ihre Web-Projekte! Zeigen Sie Ihren Website-Besuchern, dass Datensicherheit bei Ihnen höchste Priorität hat:

- ✓ Georedundanz
- ✓ DDoS Protection
- ✓ SSL-Zertifikat inklusive
- ✓ Zertifizierte Rechenzentren

Internet made in Germany

1 TEST 30 TAGE AUSPROBIEREN

1 MONAT VERTRAGS-LAUFZEIT

1 ANRUF EXPERTEN-RAT

DE: 02602/9691
AT: 0800/100668

1&1

1und1.info

Weitere Informationen finden Sie auf www.1und1.info. 1&1 Internet SE, Elgendorfer Straße 57, 56410 Montabaur.

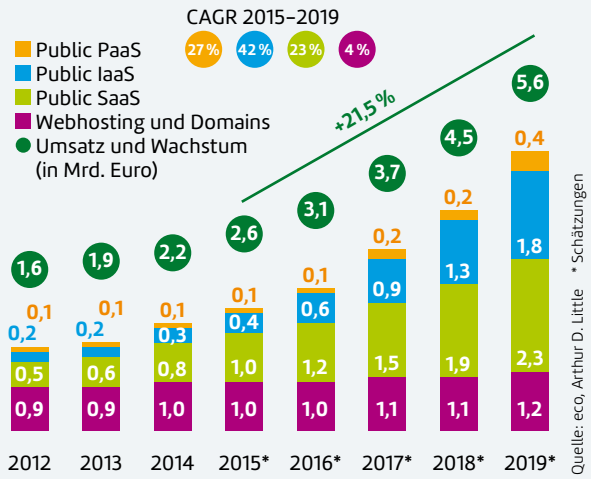
Studie

Webhosting- und Domain-Dienstleister

Die Sparte Webhosting und Domains der deutschen Internetwirtschaft mit prominenten Vertretern wie 1&1, Strato und Host Europe hat im Jahr 2015 einen Umsatz von einer Milliarde Euro erwirtschaftet. Bis 2019 wird dieser im Kerngeschäft um jährlich durchschnittlich vier Prozent auf 1,2 Milliarden Euro wachsen. Doch im weitgehend gesättigten Markt von Websites hat der Verteilungskampf um die größten Stücke des Kuchens längst begonnen, zeigen eco – Verband der Internetwirtschaft e. V. und Arthur D. Little in ihrer gemeinsamen aktuellen Studie »Die deutsche Internetwirtschaft 2015 – 2019«.

»Das Geschäft ist stabil und dabei sehr profitabel«, so Lars Riegel, Principal bei Arthur D. Little. »Mit 30 bis 40 Prozent hat Deutschland eine der höchsten Bruttomargen in Europa.«

Auch in anderer Hinsicht nimmt Deutschland einen der europäischen Spitzenplätze ein: Knapp 70 Prozent der deutschen



Umsatz und Wachstum im Bereich Services & Applications

Unternehmen haben eine eigene Webseite und in Deutschland sind mehr als 20 Millionen Domains registriert, davon rund 70 Prozent .de-Domains.

Seit etwa zwei Jahren werden in größerer Zahl neue Top Level Domains in den Markt eingeführt. Zu den neuen Endungen zählen beispielsweise .berlin oder .cloud. Einen großen Effekt werden sie auf den Umsatz der Webhosting- und Domain-Anbieter aber nicht haben.

Da durch die hohe Marktpenetration im Kerngeschäft der Webhosting- und Domain-Anbieter nur mit einem leichten Wachstum zu rechnen ist, lohnt sich der Blick über den Tellerrand. Zahlreiche Unternehmen haben bereits erkannt, dass sich abseits des Kerngeschäfts deutlich größere Wachstumspotenziale auftun. Einen großen Vorteil bringen sie dabei schon mit: Zu vielen Unternehmen und Privatpersonen mit Webseiten und Domains bestehen bereits Kundenbeziehungen mit regelmäßigem Geldfluss für die Dienstleistungen. Somit liegt es nahe, den Bestandskunden zusätzliche Services anzubieten. Dabei sind unterschiedliche Strategien zu beobachten.

www.eco.de/internetstudie

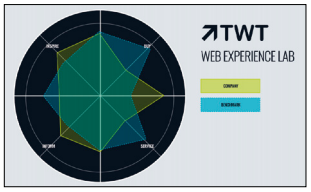
Web Experience Lab von TWT

Website-Check für alle Branchen

Die Agenturgruppe TWT Interactive hat einen individuellen Web-Performance-Check für Unternehmens-Websites entwickelt. Der Web Experience Lab analysiert Webauftritte anhand relevanter Erfolgs-Kriterien und bewertet den Ist-Zustand einer Website. Interessierte Unternehmen erhalten so einen wertvollen Marktvergleich sowie einen Überblick ihrer Potenziale und Stärken.

TWT bietet zudem für ausgewählte Branchen Benchmarks an, denen die eigenen Analyseergebnisse gegenübergestellt werden können, um eine gute Grundlage für eine anschließende Gap-Analyse zu erhalten.

»Betrachtet man heute erfolgreiche digitale Angebote über alle Branchen hinweg, so haben alle gemein, dass sie ein ausbalanciertes Verhältnis zwischen Kaufanreizen, Unternehmensinformationen, Storytelling und Service-Angeboten aufweisen und keine Dimension die andere überlagert und schwächt. Unsere Methode hilft Unternehmen dabei, schnell und effektiv Potenziale im eigenen digitalen Angebot aufzuzeigen und die eigene Präsenz



Web Experience Lab: Individueller Web-Performance-Check für Unternehmens-Websites

mit dem direkten Marktumfeld abzugleichen«, erklärt Marc Heydrich, Geschäftsbereichsleiter Experience Platforms bei TWT.

www.twt.de

LeaseWeb-Umfrage

Deutsche Wirtschaft verlangt deutsche Datenzentren

Die große Mehrzahl der deutschen Unternehmen besteht darauf, dass ihre Daten ausschließlich in deutsche Datenzentren ausgelagert werden. Das ist die Quintessenz einer Umfrage des Hosting-Netzwerks LeaseWeb unter 100 überwiegend mittelständischen IT-Unternehmen. Mehr als zwei Drittel (68 Prozent) der kontaktierten Firmen legen demnach höchsten Wert darauf, dass ihre Daten in einem Rechenzentrum

Deutsche Unternehmen bevorzugen bei ihren Cloud-Aktivitäten in erster Linie deutsche Datenzentren

verarbeitet werden, das dem deutschen Datenschutz unterliegt.

»Die nationale Fokussierung beim Datenschutz hängt damit zusammen, dass der neue EU-US Privacy Shield leider nicht die erhoffte Rechtssicherheit für die deutsche Wirtschaft bei der Auslagerung von Daten in die USA gebracht hat«, analysiert Benjamin Schönfeld, Geschäftsführer von LeaseWeb Deutschland. Das deutsche Unternehmen gehört zwar zum weltweiten Hosting-Netzwerk LeaseWeb, bietet nach eigenen Angaben jedoch Hosting-Dienste an, bei denen der Datenschutz nach deutschem Recht gewährleistet ist. »Wir bieten europäischen und speziell deutschen Unternehmen ein Hosting-Angebot an, bei dem die Daten garantiert auf euro-

päischen und auf Wunsch ausdrücklich nur auf deutschen Servern gehalten werden«, versichert Benjamin Schönfeld.
www.leaseweb.de

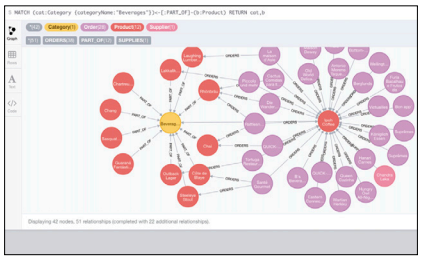
Neo Technology

Neo4j 3.0: Flotter und einfacher zu bedienen

Neo Technology, Hersteller der Graphdatenbank Neo4j, hat die Verfügbarkeit von Neo4j 3.0 angekündigt.

Das neue Release will Graphdatenbanken für den breiten Markt etablieren und setzt dafür auf enorme Skalierbarkeit, neue Treiber für Programmiersprachen und weitere entwicklerfreundliche Features.

Jim Webber, Chief Scientist von Neo Technology: »Jeder Aspekt der Architektur wurde neu überarbeitet. Ziel war es, das Entwickeln einer Anwendung mit Neo4j so effektiv und intuitiv wie möglich zu gestalten und gleichzeitig die Skalierbarkeit deutlich zu erhöhen – bis hin zur Verarbeitung von mehreren Billionen von Knoten und Kanten.«



Von der Graphdatenbank Neo4j gibt es eine neue Version

Neo4j 3.0 wird von einem neuen, binären Protokoll (Bolt) unterstützt. Zudem bietet es jetzt offizielle Treiber für Java, .NET, JavaScript und Python. Laut Hersteller eignet sich Neo4j 3.0 dank seiner Leistungsfähigkeit bei Echtzeit-Abfragen für umfangreiche Echtzeit-Anwendungen.

Zu den Highlights von Neo4j 3.0 zählen eine neue Storage-Engine für höhere Skalierbarkeit bei gleichbleibend hoher Performance sowie die Unterstützung von Java Stored Procedures. Diese ermöglicht zum Beispiel das Hinzufügen von Graphalgorithmen oder die Bereitstellung dynamischer Metainformationen in Kombination mit Bolt und Cypher.

Neben den neuen Features in Neo4j bringt Neo Technology einen neuen cloudbasierten Synchronisationsdienst auf den Markt: Neo4j Browser Sync. Der kostenlose Service erweitert das Neo4j-Angebot an frei zugänglichen Entwickler-Tools und ermöglicht es Anwendern, gespeicherte Skripts und Einstellungen über Arbeitsplätze hinweg zu synchronisieren.
http://neo4j.com

Jenkins 2.0

Continuous Delivery mit nativem Pipeline-Support

Das Jenkins-Projekt, eine Community von Anwendern von Open Source Jenkins, hat die Verfügbarkeit von Jenkins 2.0 bekannt gegeben. Es ist das erste Haupt-Release nach zehn Jahren mit 655 wöchentlichen Releases. Verschiedene Jenkins-Kernfunktionen wurden überarbeitet und aktualisiert, unter anderem gibt es nun verbesserten Support für Continuous-Delivery-Pipelines, eine transparente Darstellung der Delivery-Stadien und Erweiterungen, mit denen neue Benutzer noch schneller produktiv arbeiten können.

»Durch die Bereitstellung von Pipeline-as-Code-Funktionen und durch Visualisierung des Software-Pipeline-Status in Jenkins erleichtert mir Jenkins 2.0 die Definition sowohl einfacher wie auch komplexer Pipe-

Internetagentur-Ranking

Honorarumsätze steigen um 18 Prozent

Das Wachstum der Full-Service-Digitalagenturen in Deutschland setzt sich weiter fort, wie der Report »Internetagentur-Ranking 2016« des Bundesverbandes Digitale Wirtschaft (BVDW) zeigt.

Die Full-Service-Digitalagenturen erwirtschafteten im Geschäftsjahr 2015 Honorarumsätze von 1,432 Milliarden Euro – ein Plus von fast 18 Prozent im Vergleich zum Vorjahr (1,218 Milliarden Euro). Auch die Anzahl der Arbeitsplätze in der Branche stieg weiter an: Mit mehr als 14.300 festangestellten Mitarbeitern ist die Beschäftigtenzahl gegenüber 2014 (rund 12.800)

Anke Herbener ist Vorsitzende des Fachkreises Full-Service-Digitalagenturen im BVDW

um 12 Prozent gewachsen. Der durchschnittliche Pro-Kopf-Umsatz ist mit 89.162 Euro knapp fünf Prozent höher als noch im Vorjahr (85.219 Euro). Allein die dem Umsatz nach 50 größten Agenturen setzten im vergangenen Jahr rund 1,079 Milliarden Euro um (2014: 0,861 Milliarden Euro).

Anke Herbener (DigitasLBi), Vorsitzende des Fachkreises Full-Service-Digitalagenturen im BVDW, erklärt: »Die Full-Service-Digitalagenturen setzen ihren beeindruckenden Wachstumskurs fort und zählen zu den wichtigsten Treibern der digitalen Wirtschaft in Deutschland. Die digitale Transformation erweitert das Themenportfolio massiv. Dabei stehen uns viele Digitalisierungsprojekte in den Unternehmen noch bevor. Hier wird ein Potenzial frei, das den Full-Service-Agenturen mit ihrer in dieser Kombination einzigartigen, ganzheitlichen Beratungs-, Konzeptions- und Umsetzungskompetenz in den kommenden Jahren ein stabiles Wachstum sichern wird.«

Die beiden ersten Plätze belegen weiterhin die Agenturen Plan.net und UDG United Digital Group. Darauf folgt Team neusta, im Vorjahr noch auf Platz vier. Neu in den Top 10 sind Publicis Pixelpark, Reply – Digital Experience, MGM Technology Partners und Diva-e Digital Value Enterprise GmbH. Sinner-Schrader AG, C3 Creative Code und Content GmbH und Valtech zählen wie auch 2014 zu den nach Umsatz zehn größten Full-Service-Digitalagenturen.
www.agenturranking.de

App Store vs. Play Store

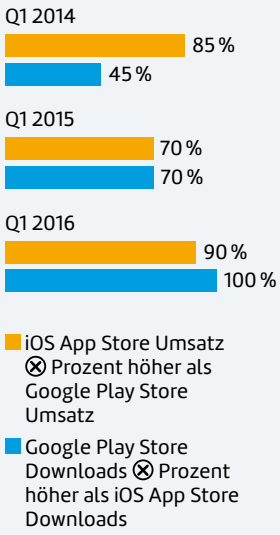
Google hat bei den Downloads die Nase vorn

Im ersten Quartal 2016 erwirtschaftete Apples App Store laut einer Studie von App Annie 90 Prozent mehr Umsatz als Konkurrent Google mit seinem Play Store. Und das, obwohl Android-Nutzer dort im selben Zeitraum doppelt so viele Programme heruntergeladen haben.

Das liegt zum einen wohl daran, dass User, die sich ein Smartphone für 600 Euro leisten, mutmaßlich auch mehr Geld für Apps ausgeben als der Käufer eines Budget-Geräts mit Android-Betriebssystem. Viele Android-Apps finanzieren sich zudem über eingeblendete Werbung. Google verdient zwar auch dabei in den meisten Fällen mit, diese Zahlen werden aber nicht dem Umsatz des Play Stores zugerechnet.

www.appannie.com

App Store: mehr Umsatz, weniger Downloads



Performance von iOS App Store und Google Play Store

Quelle: Statista

Verschiedene Jenkins-Kernfunktionen wurden in der Version 2.0 überarbeitet und aktualisiert

lines«, sagte Shashikant Jagtap, DevOps and Automation Engineer, Jenkins-Nutzer und Leiter des Jenkins Area Meetup in London. »Diese neuen Features werden es uns letztendlich ermöglichen, eine stabile Automatisierung des Continuous-Delivery-Prozesses zu realisieren und Software in noch höherer Qualität schneller als je zuvor zu produzieren.«

https://jenkins.io/2.0

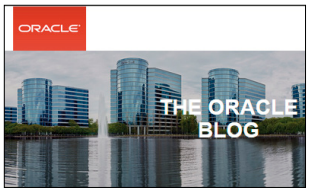
DOAG

Oracle bekommt neues, vereinfachtes Vertragswesen

Im Rahmen der Collaborate 16 hat Jeb Dasteel, Chief Customer Officer Oracle Corporation, Stefan Kinnen als Repräsentant der DOAG zu einem Executive Roundtable eingeladen. Thema war die Ankündigung des Programms »Oracle Accelerated Buying Experience«, mit dem der gesamte Einkaufsprozess und das zugrunde liegende Vertragswerk komplett überarbeitet und deutlich vereinfacht werden soll.

Oracle hat laut eigener Aussage erkannt, dass sein Vertragswerk zu umständlich und im Zeitalter des Cloud Computing nicht mehr zeitgemäß ist. Der Umfang der Verträge führt teilweise zu mehrfachem iterativem Austausch mit den Kunden, was den eigentlichen Abschluss um mehrere Wochen oder Monate verzögert. Das sei nicht länger zu akzeptieren, sagt auch die CEO-Spitze von Oracle.

»Wenn andere Cloud-Anbieter ihr Vertragswerk auf wenigen Seiten zusammenfassen und weltweit gültig vereinfachen, kann Oracle das auch«, fasst Jeb Dasteel die Motivation zusammen und betont, dass er in seiner langen Oracle-Karriere selten eine Bereitschaft zu so tiefgreifenden Veränderungen erlebt habe. Bis hin zum



Oracle hat das Programm »Oracle Accelerated Buying Experience« angekündigt

1-Click-Service für Abschlüsse bis etwa 100.000 US-Dollar werde alles einfacher.

»Die DOAG begrüßt diesen Schritt von Oracle, zumal in der Vergangenheit immer wieder Fragen in Bezug auf europäische beziehungsweise nationale Rechtsprechung diskutiert wurden«, bewertet Stefan Kinnen das Vorhaben.

www.doag.org

Deutsche Telekom

Whitepaper: Sicherheit im industriellen IoT

Die Vernetzung und Automatisierung industrieller Prozesse birgt neue Herausforderungen in puncto IT-Sicherheit. Die Deutsche Telekom erklärt in ihrem neuen Whitepaper, wo die Risiken liegen und wie sich Unternehmen dagegen wappnen können.

Verschiedene Bausteine einer industriellen IoT-Lösung sowie die notwendige Öffnung von Netzen und Systemen für Kunden, Lieferanten und Partner machen Sicherheit im industriellen Internet zu einer komplexen Aufgabe.

Wie das Whitepaper zeigt, gibt es dafür keine Patentlösung. Hundertprozentige Sicherheit ist ebenso realitätsfern wie der Wunsch, Sicherheitsmaßnahmen zu implementieren und das Thema danach zu den Akten zu legen.

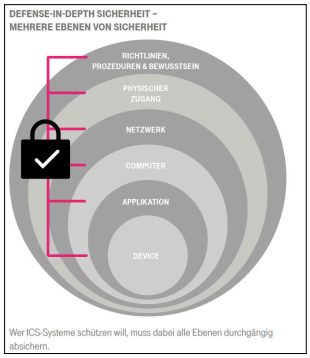
Ein adäquates Sicherheitsmanagement beginnt schon bei der Auswahl von Anlagen, Geräten, Systemen und Netzwerkkomponenten.

Hier sollten Unternehmen darauf achten, ob der Hersteller Sicherheitsaspekte in das Produkt integriert hat. Das gilt auch für den Bezug von Konnektivitäts- und Cloud-Diensten.

Um sich darüber hinaus zu wappnen, empfiehlt die Telekom Unternehmen ein umfassendes Sicherheitskonzept nach dem Defense-in-Depth-Ansatz zu entwickeln.

Dabei wird die IT-Architektur in verschiedene Schichten aufgeteilt und mit passenden Sicherheitsmaßnahmen versehen. Gelingt es einem Angreifer, die Barrieren einer Schicht zu überwinden, steht er vor der nächsten verschlossenen Tür.

www.telekom.com



In einem Whitepaper bietet die Telekom Infos zur Sicherheit im industriellen IoT

The Adaptive Workplace – Arbeiten im 21. Jahrhundert

Wie sich die Arbeitsumgebung den Menschen anpasst

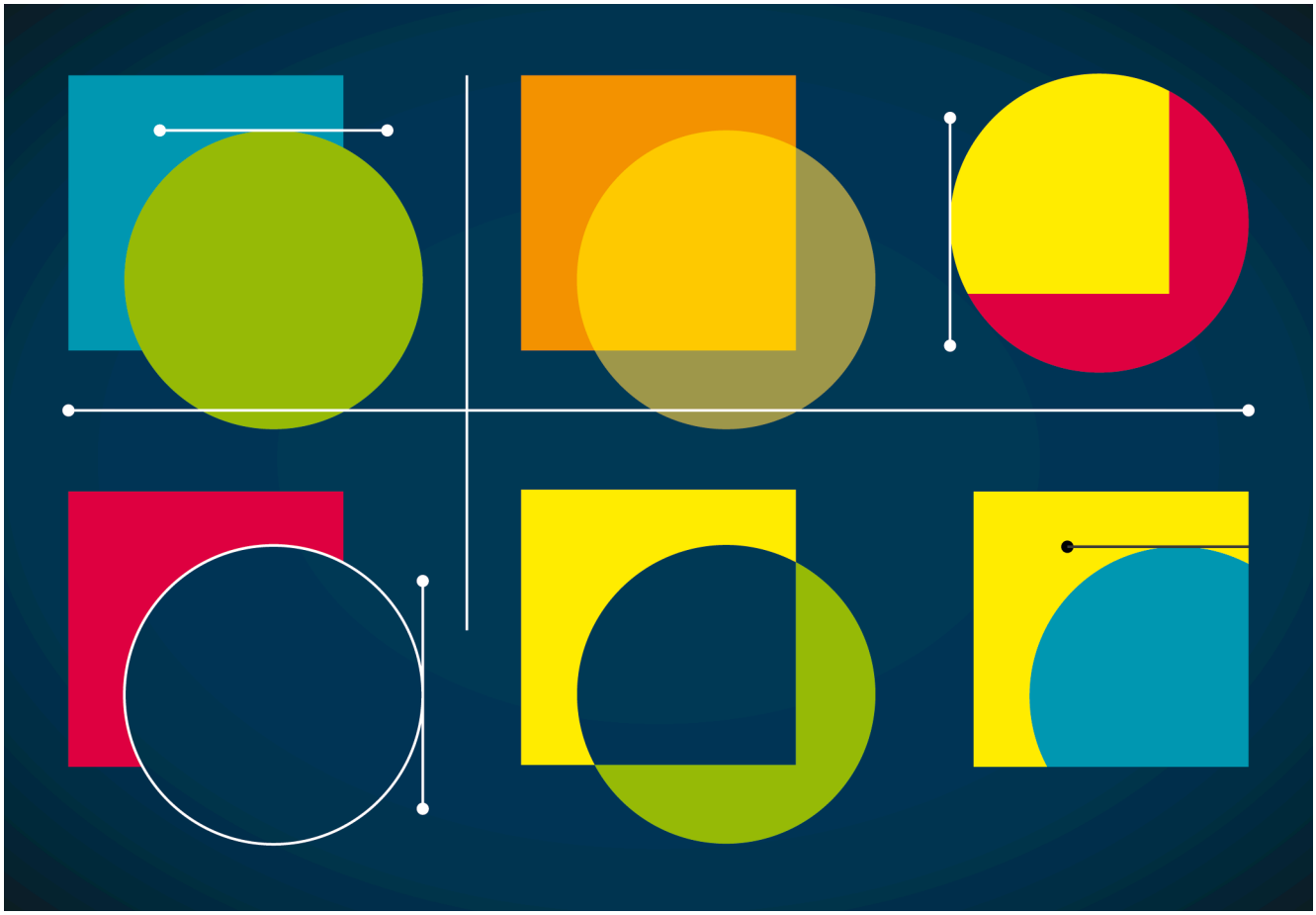
GRATIS Whitepaper

Lesen Sie im kostenfreien Whitepaper von CITRIX® und crisp RESEARCH

- Die Organisation im Wandel
- Der Arbeitsplatz im Wandel
- Ideen und Umsetzung
- Nicht ohne meine IT
- Gestaltungskriterien und Wertbeitrag
- Best Practises und Herausforderungen
- Ausblick und Empfehlungen

Jetzt kostenlos downloaden:
http://digital.internetworld.de/citrix-wp-crisp-wump/

Ein Service von:
INTERNET WORLD Business
Ihr Whitepaper bei uns?
Anfragen an sales@nmg.de
oder Tel. 089 / 7 41 17 – 124.



CANVAS UND WEBGL AUF CANVAS

Grafik-Frameworks und -Bibliotheken

Das HTML5-Element <canvas> bildet die Basis innovativer JavaScript-Lösungen.

Das moderne Web möchte den Benutzer nicht nur mit üppiger Grafik und bewegungsreichen Animationen verwöhnen, sondern durch ausgefallene Interaktivität engagieren. Anzeigen wollen angeklickt werden und 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 (Bild 1) immer öfter anzutreffen sind (Bild 2).

Noch nie zuvor standen Webentwicklern so viele verschiedene Technologien zur Verfügung, um diese Ziele zu erreichen. Visuell ansprechend, unterhaltsam und engagierend: Eine Webapplikation kann heute all diese Merkmale besitzen, ohne den Browser in die Knie zu zwingen. Das Ableben von Adobe Flash hat die Vielfalt der grafischen Möglichkeiten des Webs kaum wirklich beeinträchtigt. Die beiden führenden Technologien, HTML5 mit dem *<canvas>*-Element in Kombination mit CSS3 und gegebenenfalls geeigneten JavaScript-Bibliotheken, SVG und WebGL bieten eine enorme Vielfalt an technischen Möglichkeiten. Webentwickler haben somit die Qual der Wahl.

Das *<canvas>*-Element ist ein Bestandteil der HTML5-Spezifikation und ermöglicht Grafikmanipulationen in JavaScript. Es stellt im Webbrowser eine rechteckige Leinwand bereit, auf der unter anderem mittels JavaScript gezeichnet werden kann.

Der visuelle Inhalt einer so erzeugten Leinwand entsteht in JavaScript, zum Beispiel so:

```
var canvas = document.getElementById("canvasName");
var ctx = canvas.getContext("2d");
```

```
ctx.fillStyle = "rgb(25,130,97)";
ctx.fillRect(10, 10, 100, 100);
```

Konzeptionell ähnelt *<canvas>* dem * *-Element. Während es sich allerdings bei dem * *-Element um einen Container für andere HTML-Tags handelt, fungiert das *<canvas>*-Element als ein Container für Pixel. In einem *<canvas>*-Element lassen sich unter anderem Grafiken zeichnen, Fotos komponieren, Animationen gestalten und Videoclips bearbeiten. Canvas unterstützt jedoch im Gegensatz zu SVG keinen Text, es lassen sich darin ausschließlich Pixel manipulieren.

Auf Hochglanz poliert: Dieses detailreiche 3D-Modell eines Autos in einer abbröckelnden Garage demonstriert die bemerkenswerten Grafikfähigkeiten aktueller Webbrowser (Bild 1)

Hohe Performance

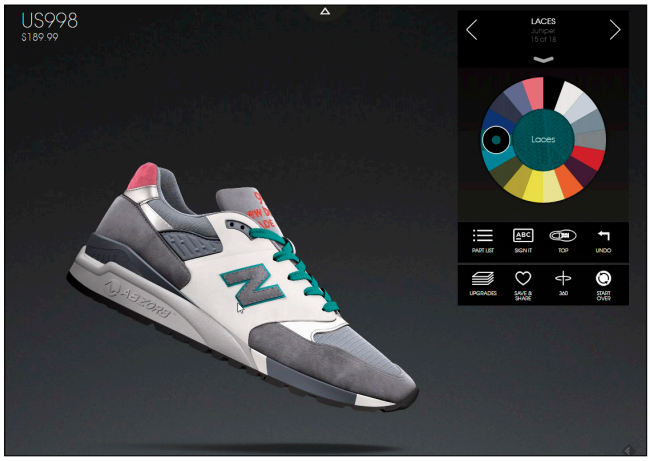
Canvas bietet dem Entwickler vor allem Simplizität und eine hohe Performance, hat jedoch auch einige Nachteile. Gezeichnete Objekte, darunter Vektorobjekte, werden in die *<canvas>*-Bitmap hineingerendert und lassen sich daher nicht an einen Event Handler anknüpfen. Klickt der Benutzer ein Grafikelement an, löst dies kein natives Ereignis aus. Der Entwickler kann diese Problematik mit JavaScript umgehen, indem er dem gesamten *<canvas>*-Element einen Ereignis-Handler zuweist und die Pixelposition des Mausklicks auswertet.

Die Ereignisauswertung im Browser läuft dann aber nicht nativ. Auch kann der Webbrowser die Bildschirmkoordinaten nicht nativ in das Weltkoordinatensystem der Leinwand umwandeln. Der Entwickler kann eine Transformation codieren, die ein Koordinatensystem in das jeweils andere umrechnet. Diese Konversion muss allerdings bei jeder Größenänderung des *<canvas>*-Elements aufs Neue erfolgen.

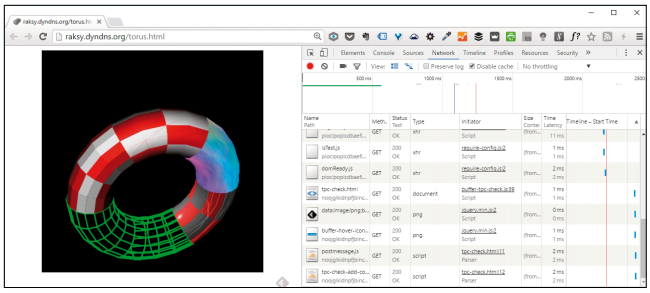
Beim Einsatz des *<canvas>*-Elements in einem elastischen Layout müssen die Gleichungskoeffizienten der Transformation sogar bei jeder Änderung der Fenstergröße neu berechnet werden. Der Browser kann die Canvasgröße auf der Basis der aktuellen Fenstergröße nicht nativ berechnen; der Entwickler muss also dafür sorgen, dass bei jeder Fenstergrößenänderung die erforderliche Umrechnung vorgenommen wird. Zum Animieren von Objekten auf der Canvas muss diese zudem bei jedem Bewegungsschritt erneut gezeichnet werden. Dennoch lassen sich auf der Canvas mit reinem JavaScript auf einer Webseite durchaus interessante Effekte erzielen (Bild 3).

Zum Zeichnen komplexer Grafiken bietet das Canvas-API ein Pfad-API mit Befehlen, die bei SVG Anleihen machen. Für jedes Liniensegment müssen Sie beim Einsatz von Canvas einen erneuten API-Aufruf tätigen.

Es gibt nur eingeschränkte Möglichkeiten, die Position der Maus auf der Leinwand zu erfassen, um sie als ein Ereignis zu interpretieren. Der Entwickler muss für jedes visuelle Objekt Klick-Koordinaten ermitteln *(mouseX, mouseY)*, um diese Auswertung überhaupt möglich zu machen, denn es liegen ja keine Informationen zu den Grafikelementen, sondern lediglich ein Bitmap der Leinwand in ihrem jeweils aktuellen Zustand vor. Spezialisierte Bibliotheken können daher beim Interpretieren der Ereignisse wertvolle Hilfe leisten. ▶



Verbraucherfreundlich: Diese interaktive Webshop-Applikation von Newbalance.com auf der Basis von EazelJS erlaubt es dem Käufer, jeden einzelnen Wunschartikel in der Bestellung ganz nach eigenen Vorstellungen zu gestalten (Bild 2)



Auf der Leinwand malen: Grafische Effekte, die sich mit reinem JavaScript realisieren lassen, beinhalten Gouraud-Shading und Environment-Mapping, werden jedoch auf der Canvas ohne GPU-gestützte Hardwarebeschleunigung gerendert (Bild 3)

Der eigentliche Trick beim Einsatz von *<canvas>* besteht darin, das sprichwörtliche Rad nicht neu zu erfinden: Wer ein geeignetes Framework zur Hilfe ruft, verschafft sich einen sehr großen Produktivitätsvorsprung.

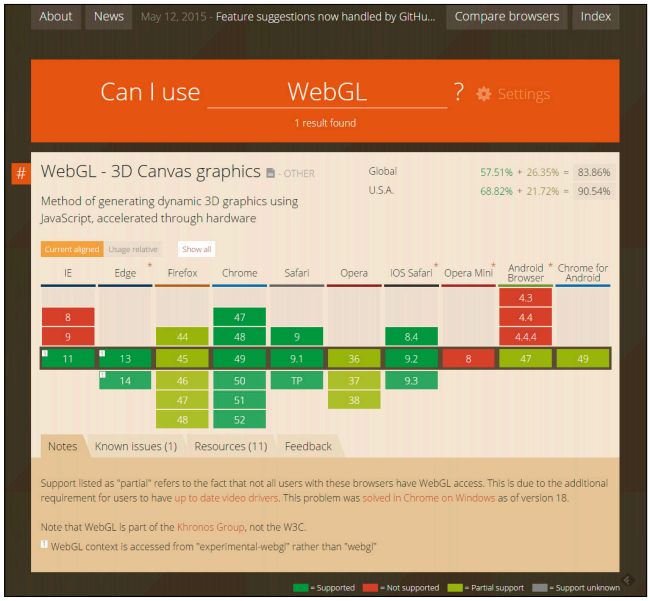
SVG als Alternative

Wer mehr Flexibilität benötigt, als sie das *<canvas>*-Element bieten kann, greift oft zu SVG. Bei SVG handelt es sich um einen offenen Standard für zweidimensionale Grafik mit Unterstützung für Interaktivität und Animation. Zu den beliebtesten SVG-Frameworks zählt Raphaël.

SVG-Grafiken und ihr Verhalten werden durch XML-Dateien beschrieben; die Leinwand einer SVG-Grafik bezeichnet man als ihren Viewport. Da jedes Grafikobjekt Teil des DOM ist, können ihm Event Handler zugeordnet werden, die dann im Browser nativ ausgewertet werden. Der Browser ist auch selbst für die Auswertung der SVG-Grafik im Kontext eines elastischen Layouts verantwortlich. Bei einer hohen Komplexität leidet allerdings die Leistung. Beim Animieren von Objekten in einer SVG-Grafik müssen nur die animierten Objekte aktualisiert werden. Die Bildschirmkoordinaten lassen sich in das Spielkoordinatensystem mit Hilfe einer Transformationsmatrix übersetzen.

SVG kann zwar (auch) pixelbasierte Grafiken darstellen, aber es ist erstens auf das Zeichnen von Vektoren hin optimiert, und zweitens erfolgt dies ohne jegliche Hardwarebeschleunigung. Wenn es darum geht, interaktive Erlebniswelten in virtueller Realität ins Leben zu rufen, greifen sowohl SVG als auch das reine Canvas-Element aufgrund fehlender Hardwarebeschleunigung leider zu kurz. Eine Lösung bietet WebGL, eine Technologie zum Rendern der Ausgabe für das *<canvas>*-Element in HTML5 direkt auf der GPU (Tabelle 1).

Bei WebGL (Web Graphics Library) handelt es sich um ein lizenzfreies Low-Level-Zeichnungs-API für dynamische Webapplikationen auf der Basis von OpenGL ES 2.0. Es gehört nicht zur offiziellen HTML5-Spezifikation des W3C, sondern wurde von den Browserherstellern zum De-facto-Standard gekürt. WebGL ist plattformunabhängig; es erlaubt die Erstellung anspruchsvoller und zugleich leichtgewichtiger 2D- und 3D-Animationen für das Web und für mobile Apps, Online-Spiele, Datenvisualisierungen, CAD, Augmented Reality et cetera. WebGL-Animationen werden in einem *<canvas>*-Element unter Angabe von WebGL als Kontext ausgegeben. Dadurch laufen sie einfach ohne ein Plug-in in einem Browser (Bild 4). Anders als im Fall von reinem Canvas entsteht die eigentliche Visualisierung jedoch durch Low-Level-Zugriffe auf die GPU.



Auf der Höhe der Zeit: Die Unterstützung von WebGL ist mittlerweile weit verbreitet; das traurige Schlusslicht unter den Desktop-Browsern ist Firefox (Bild 4)

Das Besondere an WebGL gegenüber anderen Grafiktechnologien ist die rekordverdächtigte Grafikleistung, die sich dank der direkten Adressierung des Grafikprozessors des Endgeräts auch bereits auf Mobilgeräten in halbwegs aktuellen Browsern erzielen lässt (Bild 5).

WebGL bricht mit Konventionen klassischer Webprogrammierung, indem zusätzlich zu JavaScript eine 3D-fähige Rendering-Sprache – OpenGL Shading Language – zum Einsatz kommt und die GPU des Zielgeräts direkt adressiert.

OpenGL Shading Language

Die JavaScript-Komponente zeichnet für das Laden von Bildern verantwortlich, definiert Farben und beschreibt Objekte. Die GLSL-Komponente übersetzt diese Daten für die Ausführung auf der GPU unter Verwendung von sogenannten Shaderprogrammen. Die Resultate werden im Kontext eines Canvas-Elements als 2D-Bitmaps ausgegeben. Das Resultat ist Spitzenleistung beim Rendern komplexer Szenen auf nahezu beliebigen Endgeräten ohne Browser-Plug-ins.

Ein Shaderprogramm setzt sich aus den sogenannten Shadern zusammen, speziellen Funktionen in GLSL, die Sie entweder selbst programmieren müssen oder für Ihre Applikation unter Verwendung von einer WebGL-Bibliothek bereitstellen. Es kommen jeweils immer zwei dieser Shader zum Einsatz: ein sogenannter Vertex-Shader und ein sogenannter Fragment-Shader.

Der Vertex-Shader kommt in der ersten Phase der Verarbeitung der Grafikdaten durch die OpenGL-Pipeline zum Zuge; er hantiert mit Eckpunkten eines Objekts im 3D-Raum und übersetzt diese in das Screen-Koordinatensystem:

```
precision highp float;
attribute vec3 position;
attribute vec2 uv;
uniform mat4 SpielWeltProjektion;
varying vec2 vUV;
void main(void) {
  gl_Position = SpielWeltProjektion *
  vec4(position, 1.0);
  vUV = uv;
}
```

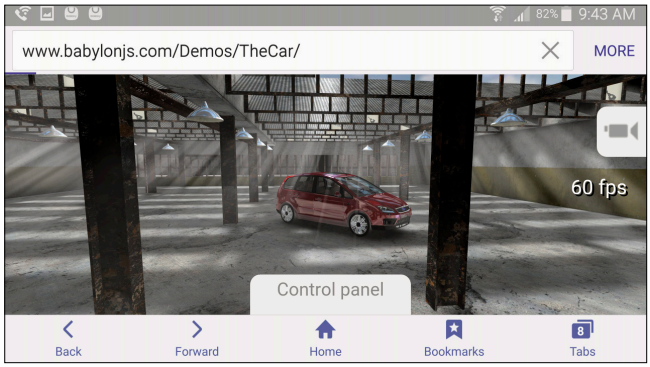
Die zugehörigen Daten (gesichert in dem sogenannten Vertex-Buffer) werden anhand von einer oder mehreren Matri-

Tabelle 1: Grafik-APIs des Webs und ihre Anwendungsszenarien

	Canvas	SVG	WebGL/Canvas
Kontext	2D oder 3D	2D	Canvas (3D in 2D)
internes Szenendiagramm	nein (lässt sich in JavaScript nachbilden)	ja	ja
Ausgabe	pixelbasiert (Bitmap)	vektorbasiert (Pfade)	pixelbasiert (Bitmap)
DOM-Integration	einzelnes HTML-Element *<canvas>*	mehrere Elemente, die Teil des DOM (Dokumentobjektmodells) sind	einzelnes HTML-Element *<canvas>*
Inhalte modifizierbar mittels	JavaScript u. a.	JavaScript und/oder CSS	JavaScript und GLSL (OpenGL Shading Language)
Ereignismodell für Benutzerinteraktionen	präzises Ereignismodell für Benutzerinteraktionen (x,y)	allgemeines Ereignismodell für Benutzerinteraktionen *(rect, path)*	integriert mit dem Ereignisauswertungssystem des Webbrowsers
optimale Leistung	bei einer kleineren Leinwandgröße (mit geringerer Pixelzahl) und/oder einer sehr hohen Objektanzahl	unabhängig von der der Leinwandgröße (da vektorbasiert) bei einer kleineren Objektanzahl	in Abhängigkeit von der Art, der Anzahl und der Präzision von Shadern, der Bittiefe und der Auflösung von Texturen sowie der Rendering-Auflösung von Canvas
Unterstützung für Text/ Barrierefreiheit	nein/nein	ja/ja	nein/nein
ideale Inhaltstypen/ Anwendungsszenarien	komplexe Szenen, Grafikfilter, Effekte wie Spiegelungen, Lichtstrahlverfolgung, mathematisch berechnete Animationen; Echtzeitdaten, Pixelersetzung auf der Basis von Farberkennung (Greenscreen für Video)	Inhalte, die mit einer hohen Detailtreue unabhängig von der Darstellungsgröße wiedergeben werden müssen (Bau-, Konstruktions- und Grundrisspläne, Karten, Diagramme et cetera)	Spiele, Simulationen, 3D-Modelle, Erlebniswelten, engagierende Werbebanner, die auf möglichst vielen Geräten ohne Browser-Plug-ins laufen sollen

ces transformiert, gerastert und Pixel für Pixel an den Fragment-Shader übergeben. Der Fragment-Shader erfasst eine Repräsentation der Daten in der abschließenden Phase der OpenGL-Pipeline: nach dem Rastern der Primitive und vor der Übergabe der Daten an den Framebuffer:

```
precision highp float;
varying vec2 vUV;
```



Abgefahren: Eine virtuelle 3D-Welt aus BabylonJS in WebGL bei einer Framerate von 60 fps auf einem Smartphone (Bild 5)

```
uniform sampler2D textureSampler;
void main(void) {
  gl_FragColor = texture2D(textureSampler, vUV);
}
```

In dieser Phase werden die Farbwerte der Vertices interpoliert, die Texturen angewendet und weitere Manipulationen mit Hilfe des Shaders umgesetzt. Der Vertex- und der Fragment-Shader müssen miteinander verbunden werden, sodass sie ein Shaderprogramm bilden, das dann die Daten entlang der gesamten OpenGL-Grafik-Pipeline verarbeitet. Eine typische WebGL-Applikation besteht aus vielen solchen Shaderprogrammen, die verschiedene Elemente der Szene zeichnen. WebGL-Animationen können andere Seitenobjekte überlagern oder auch im Hintergrund anderer Seitenelemente ablaufen (zum Beispiel in einem in der z-Achse angeordneten * *-Element).

WebGL in einer Webapplikation

Um WebGL in einer Webapplikation zu implementieren, sind einige Schritte umzusetzen. Das HTML-Dokument muss ein *<canvas>*-Element beinhalten. Das folgende Listing illustriert ein *<canvas>*-Element und den Aufruf der Initialisierungs- ▶

funktion aus dem *onload*-Ereignis-Attribut des *<body>*-Elements heraus. Der zugehörige Rendering-Kontext wird initialisiert, sobald das Dokument ganz heruntergeladen wurde:

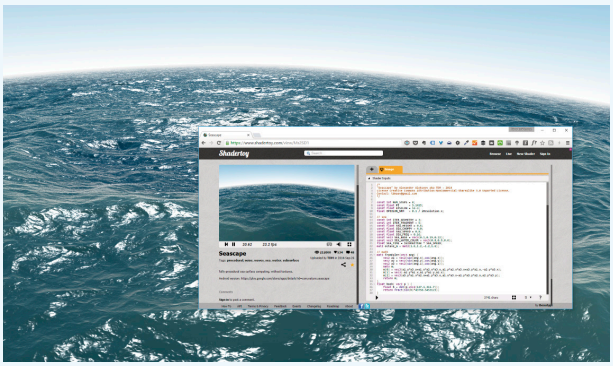
```
<body onload="start()">
  <canvas id="glcanvas"
  width="820" height="640">
    Ihr Webbrowser scheint das HTML-Element
    <code>&lt;canvas&gt;</code> nicht zu unterstützen.
  </canvas>
</body>
```

Eine Referenz auf die Canvas wird anschließend in einer Variablen erfasst:

```
var gl;
function start() {
  var canvas = document.getElementById("glcanvas");
  gl = initWebGL(canvas);
  if (gl) {
    gl.clearColor(0.0, 0.0, 0.0, 1.0);
    gl.enable(gl.DEPTH_TEST);
    gl.depthFunc(gl.LEQUAL);
    gl.clear(gl.COLOR_BUFFER_BIT |
    gl.DEPTH_BUFFER_BIT);
```

Ideen-Austauschbörse

Im Web findet sich eine Ideen-Tauschbörse für die Entwicklung von Shadern. Wer es unbedingt darauf ankommen lässt, eigene Shader zu entwickeln, findet unter der Adresse: https://www.shadertoy.com eine Ideen-Austauschbörse mit einer Goldgrube an Shader-Funktionen. Hier können Sie die Shader anderer WebGL-Entwickler im Webbrowser auf die Probefahrt nehmen, daraus Inspiration schöpfen und Ihre eigenen Kreationen der Gemeinde bereitstellen. Mit einem Klick auf das Fragezeichen rechts unterhalb eines Code-Listings können Sie relevante Teile der GLSL-ES-Spezifikation schnell nachschlagen.



Täuschend echt: Seascape, eine beeindruckende Shader-Funktion von Alexander Alekseev, rendert eine virtuelle Ozeanoberfläche so überzeugend, dass man den in Endlosschleife ablaufenden Clip für eine Videoaufnahme halten könnte (http://shadertoy.com)

```
  }
}
```

Sofern Sie die Leinwand nicht mehrmals, sondern nur einmalig referenzieren möchten, sollten Sie den Wert der zugehörigen Variablen nicht global erfassen, sondern als eine lokale Variable oder als zugehörig zu einem Objekt speichern. Die Funktion *initWebGL()* initialisiert den benötigten WebGL-Kontext für die in der Variablen *gl* erfasste Canvas.

Hat dies erst einmal geklappt, bildet die Variable *gl* eine Referenz darauf. Die Farbe wird auf Schwarz gesetzt. Danach wird der Zeichenkontext durch das Definieren von Parametern konfiguriert. Im Beispiel werden Fähigkeiten zur Probung der Tiefeneigenschaft aktiviert und das gegenseitige Verdecken von weiter weg gelegenen durch näher gelegene Objekte ermöglicht. Mittels

```
canvas.getContext(kontextTyp,
{ kontextAttribut-1: wert,
kontextAttribut-2: wert,
kontextAttribut-n: wert });
```

lässt sich der gewünschte Zeichenkontext für die Canvas anfragen. Die *canvas.getContext()*-Methode gibt im Erfolgsfall den angeforderten Renderingkontext und im Fehlerfall den Wert *null* zurück. Der String *kontextTyp* beinhaltet die gewünschte Kontextkennung. Die unterstützten Kontextattribute hängen von eben dieser Kontextkennung ab.

Sollte der Aufruf in Bezug auf den Standard-Kontext *webgl* nicht gelingen, wird der Kontext *experimental-webgl* angefordert:

```
function initWebGL(canvas) {
  gl = null;
  try {
    gl = canvas.getContext("webgl") ||
    canvas.getContext("experimental-webgl");
  }
  catch(e) {}
  if (!gl) {
    alert("WebGL konnte leider nicht initialisiert
    werden, möglicherweise weil Ihr Webbrowser diese
    Technologie nicht unterstützt.");
    gl = null;
  }
  return gl;
}
```

Sollte der Kontext nicht abgerufen werden können, nimmt die Variable *gl* den Wert *null* an und der Browser zeigt die hierfür vorgesehene Fehlermeldung an. Sollte der Kontext, in den hineingerendert werden soll, erfolgreich abgerufen worden sein, beinhaltet die Variable *gl* eine Referenz darauf und die Canvas wird mit der Farbe Schwarz gefüllt.

Ein neu erstellter WebGL-Kontext setzt seine Viewport-Dimensionen unmittelbar auf die Höhe und Breite seines Canvas-Elements, ohne CSS. Spätere Anpassungen dieser bei-

Generative Kunst

Unter dem Begriff Generative Kunst (Generative Art) versteht man eine zeitgenössische Form künstlerischen Schaffens, bei dem die gewählte Vorgehensweise mit vorab definierten Regeln – und nicht das entstehende Kunstwerk – in den Vordergrund tritt.

Das Werk entsteht durch das Abarbeiten bestimmter, vom Künstler festgelegter Anweisungen. Der Künstler kann seine Anweisungen zum Beispiel in Form von einer Software bereitstellen, die dann das eigentliche Kunstwerk generiert. Bei dem Kunstwerk kann es sich um ein Bild, eine Animation oder eine nahezu beliebige andere Form handeln.

den Eigenschaften des Canvas-Elements werden sich auf die Rendering-Auflösung des WebGL-Kontexts aber nicht mehr auswirken. Ändert der Benutzer die Fenstergröße in einem elastischen Layout auf einem Computer oder wechselt die Display-Ausrichtung auf einem Mobilgerät und löst damit eine Neuskalierung der Leinwand aus, rendert WebGL weiterhin die alte Auflösung. Jegliche Anpassungen müssen Sie daher in Ihrer Anwendung encodieren, indem Sie die *viewport()*-Funktion in Bezug auf den WebGL-Kontext aufrufen, zum Beispiel:

```
gl.viewport(0, 0, canvas.width, canvas.height);
```

Die App muss jetzt einen oder mehrere Shader mit den passenden Parametern initialisieren, um den zu verwendenden Zeichenalgorithmus zu implementieren. Die Initialisierung des Shader-Programms erfolgt in JavaScript durch den Aufruf der *createProgram()*-Funktion auf dem WebGL-Objekt:

```
function initShaders() {
  var fragmentShader = getShader(gl, "shader-fs");
  var vertexShader = getShader(gl, "shader-vs");
  shaderProgram = gl.createProgram();
  gl.attachShader(shaderProgram, vertexShader);
  gl.attachShader(shaderProgram, fragmentShader);
  gl.linkProgram(shaderProgram);
  if (!gl.getProgramParameter(shaderProgram,
  gl.LINK_STATUS)) {
    alert("Unable to initialize the shader program.");
  }
  gl.useProgram(shaderProgram);
  vertexPositionAttribute = gl.getAttribLocation
  (shaderProgram, "aVertexPosition");
  gl.enableVertexAttribArray(vertexPositionAttribute);
}
```

Dem Shader-Programm werden dann die beiden Shader mittels *attachShader()* zugewiesen und dieses wird dann selbst gelinkt. Die eigentlichen Shader-Programme werden nicht in JavaScript, sondern mit Hilfe der Sprache OpenGL ES Shading Language geschrieben und zur Laufzeit kompiliert. Die benötigten Shader-Programme werden mittels der Funktion *getShader()* typischerweise von einem *<script>*-Element in dem Webdokument anhand der zugehörigen Element-ID in eine Variable geladen:

```
function getShader(gl, id) {
  var shaderScript, theSource, currentChild, shader;
  shaderScript = document.getElementById(id);
  if (!shaderScript) {
    return null;
  }
  theSource = "";
  currentChild = shaderScript.firstChild;
  while(currentChild) {
    if (currentChild.nodeType == currentChild.TEXT_NODE)
    {
      theSource += currentChild.textContent;
    }
    currentChild = currentChild.nextSibling;
  }
```

Die Variable erfasst somit im Erfolgsfall den fertig kompilierten Shader oder im Fehlerfall den Wert *null*:

```
if (shaderScript.type == "x-shader/x-fragment") {
  shader = gl.createShader(gl.FRAGMENT_SHADER);
} else if (shaderScript.type == "x-shader/x-vertex") {
  shader = gl.createShader(gl.VERTEX_SHADER);
} else {
  // Shadertyp unbekannt
  return null;
}
```

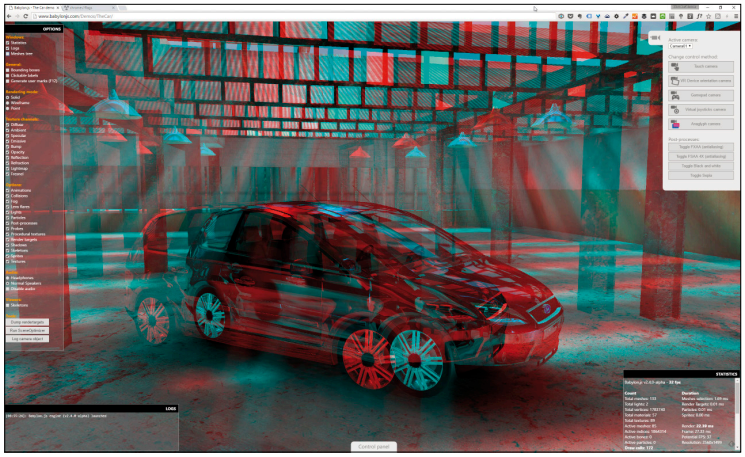
Shader-Objekte haben einen definierten MIME-Typ: *x-shader/x-vertex* (bei einem Vertex-Shader) oder *x-shader/ x-fragment* (bei einem Fragment-Shader):

```
if (shaderScript.type == "x-shader/x-fragment") {
  shader = gl.createShader(gl.FRAGMENT_SHADER);
} else if (shaderScript.type == "x-shader/x-vertex") {
  shader = gl.createShader(gl.VERTEX_SHADER);
} else {
  // Unknown shader type
  return null;
}
```

Buffer dienen dazu, die Vertex-Daten zwischenzuspeichern, zu manipulieren und an die GPU zu übergeben. Ein Buffer entsteht mittels *gl.createBuffer* und wird mittels *gl.bindBuffer* als der aktive Buffer gekennzeichnet. Mittels *gl.bufferData* werden die betreffenden Daten in den aktiven Buffer geladen. Anschließend muss die WebGL-Applikation diese Daten aus dem Buffer abholen und an die GPU übergeben.

Die Applikation muss eine oder mehrere Matrices erzeugen, um die Transformationen der gebufferten Daten auf die verfügbare Leinwand zu beschreiben. Diese undankbare Aufgabe gelingt wesentlich einfacher unter Verwendung ▶

der Sylvester-Bibliothek für Vektor- und Matrixoperationen in JavaScript sowie der Bibliothek glUtils.js zum Steuern von Sylvester, und sie entfällt komplett beim Einsatz eines Frameworks wie BabylonJS.



Tiefensimulation: Diese virtuelle 3D-Szene aus BabylonJS können Besitzer einer 3D-Brille dank der integrierten Anaglyph-Kamera in ihrer vollen stereoskopischen Pracht interaktiv erleben (Bild 6)

Zu guter Letzt kann die Applikation jetzt zeichnen und die Resultate auf die Canvas ausgeben.

Leistungsstarke Bibliotheken und Frameworks

In der Praxis entstehen WebGL-Welten meist unter Verwendung von leistungsstarken Bibliotheken und Frameworks, die dem Programmierer die schwierigen Berechnungen abnehmen. Ohne diese externen Bibliotheken müsste man sogar scheinbar triviale Manipulationen wie das Skalieren oder Drehen von Hand coden.

Auch das Beschreiben von komplexen 3D-Objekten als Abfolge von Dreiecken wird erst mit externen Bibliotheken sinnvoll möglich. Für etwas so scheinbar Simples wie eine Kugel bräuchte man ein Array mit Koordinaten für jeden Vertex-Punkt, der dann mit Hilfe trigonometrischer Daten berechnet werden müsste. Mit einer geeigneten Bibliothek können Sie eine Kugel zeichnen, indem Sie zum Beispiel wie im Fall von Three.js den Radius und die Anzahl der zu verwendenden Dreiecke angeben.

Da viele der leistungsstärksten Bibliotheken ohnehin kostenfrei verfügbar sind, ist es sinnvoll, eine geeignete Bibliothek zur Hilfe zu nehmen und aus ihren Möglichkeiten wie etwa der Bereitstellung stereoskopischer Ansichten für eine 3D-Brille mit vollen Händen zu schöpfen (Bild 6).

CreateJS

CreateJS bietet eine Sammlung von JavaScript-Bibliotheken und -Tools für die Gestaltung visuell anspruchsvoller interaktiver Inhalte in HTML5. CreateJS umfasst EaselJS, TweenJS, SoundJS und PreloadJS.

Das Framework unterstützt sowohl eine reine HTML5-Canvas als auch WebGL. Für den Umgang mit dem HTML5-Element *<canvas>* zeichnet in dieser Sammlung EaselJS verantwortlich. TweenJS vereinfacht das Tweening und das Animieren von HTML5- und JavaScript-Eigenschaften. HTML5-Audio lässt sich mit Hilfe von SoundJS steuern. Für das koordinierte, planmäßige Vorabladen der benötigten Inhalte schließlich ist PreloadJS zuständig.

Die einzelnen Bestandteile der Suite lassen sich sowohl zusammen als auch einzeln in beliebigen Kombinationen nutzen. EaselJS lässt sich ohne große Umstände in etablierte Workflows integrieren. So können Benutzer von Adobe Animate das HTML5-Document-Feature einspannen, um direkt für CreateJS auszugeben.

Die EaselJS-Bibliothek bietet einen optimierten Rendering-Modus auf der Basis einer persistenten Repräsentation der Canvas mit einem Modell der Interaktionen und Hilfsklassen zur Handhabung von 2D-Grafik in Canvas.

Beim Einsatz von EaselJS gilt es eine sogenannte Stage-Instanz zu erzeugen, die das zugehörige Canvas-Element einschließt. Anschließend fügen Sie der Liste der anzuzeigenden Objekte eine oder mehrere DisplayObject-Instanzen als Kindelemente der Szene hinzu.

EaselJS unterstützt die folgenden Inhaltstypen:

- Bilder unter Verwendung der Bitmap-Klasse,
- mittels der Klassen Shape und Graphics erzeugte Vektorgrafiken,
- animierte Bitmaps unter Verwendung der Klassen SpriteSheet und Sprite,
- einfache Textinstanzen unter Verwendung von Text,
- Behälter für andere DisplayObjects-Inhalte, die sich mittels der Container-Klasse darin einschließen lassen, um eine logische Inhaltseinheit zu bilden,
- Controller für DOM-Elemente des zugehörigen HTML-Dokuments, die unter Verwendung der *DOMElement*-Klasse eingerichtet werden.

Mit der Ausnahme der DOM-Controller feuern all diese Inhaltstypen Touch- und Maus-Ereignisse ab. EaselJS unterstützt Schweben, Drücken und Loslassen und bietet zudem ein einfaches Drag-und-Drop-Ereignismodell. Die bemerkenswerte Simplizität von EaselJS hat dem Framework viele Anhänger beschert.

Animation in EaselJS

Eine Animation in EaselJS spielt sich auf einer sogenannten Bühne ab. Eine Animation oder eine Spielebene in EaselJS bezeichnet man als eine Szene.

Die verschiedenen Akteure, die auf der Szene erscheinen, werden in einer Initialisierungsfunktion definiert, die nach dem Laden der Seite aufgerufen wird. Um EaselJS in Ihr Projekt zu integrieren, können Sie die Bibliothek aus dem CDN des Anbieters beziehen (https://code.createjs.com/easeljs-0.8.2.min.js).

Falls Sie die Objekte nicht nur zeichnen, sondern auch nahtlos animieren möchten, können Sie auch TweenJS ein-

binden: Erstellen Sie ein *<canvas>*-Element im *<body>*-Abschnitt der Webseite:

```
<canvas id="CanvasBeispiel" width="800" height="400">
</canvas>
```

In diese Canvas erfolgt die Grafikausgabe des Spiels beziehungsweise der Animation. Erstellen Sie die benötigte Initialisierungsfunktion *init()*:

```
<script>
function init() {
  // EaselJS-Code
}
</script>
```

Platzieren Sie das Skript entweder im *<head>*- oder im *<body>*-Bereich. Danach fügen Sie dem *<body>*-Element der Seite den Aufruf der Initialisierungsfunktion hinzu:

```
<body onload="init();">
```

Als Nächstes definieren Sie in Ihrem Skript die benötigte Szene. Hierzu genügt:

```
var stage = new createjs.Stage("CanvasBeispiel");
```

Ein Element der Szene entsteht mit Hilfe der passenden Klasse. So können Sie beispielsweise eine neue Instanz eines Kreises ins Leben rufen:

```
var circle = new createjs.Shape();
```

Zeichnen Sie jetzt den Kreis und füllen Sie ihn mit der gewünschten Farbe:

```
circle.graphics.beginFill
("DeepPink").drawCircle(0, 0, 50);
circle.x = 100;
circle.y = 100;
```

Fügen Sie den so erstellten Kreis mit *stage.addChild(circle);* als eine neue Kreis-Instanz der Liste anzuzeigender Objekte Ihrer Szene als ein Kindelement der Szene hinzu und aktualisieren Sie die Darstellung mit *stage.update();*.

Um diesen Kreis jetzt zu animieren, können Sie entweder weiterhin mit EaselJS arbeiten oder die Schwesterbibliothek TweenJS aufrufen. Beim Einsatz von TweenJS müssen Sie allerdings den Aufruf von *stage.update()* erst einmal entfernen, denn in TweenJS wurde diese Aufgabe anders gelöst. Fügen Sie anstelle der entfernten Zeile den TweenJS-Code aus dem nachfolgenden Listing hinzu und speichern Sie das Dokument anschließend ab:

```
createjs.Tween.get(circle, { loop: false })
.to({ x: 500 }, 1000, createjs.Ease.getPowInOut(4))
.to({ alpha: 0, y: 175 }, 500,
createjs.Ease.getPowInOut(2))
.to({ alpha: 0, y: 225 }, 100)
.to({ alpha: 1, y: 200 }, 500,
createjs.Ease.getPowInOut(2))
.to({ x: 100 }, 800, createjs.Ease.getPowInOut(2))
.to({ alpha: 0, y: 225 }, 200);
createjs.Ticker.setFPS(60);
createjs.Ticker.addEventListener("tick", stage);
```

TweenJS animiert das Objekt *circle*. Mittels *setFPS()* bestimmen Sie die Häufigkeit der Aktualisierungen der Bühne. An die Ticker-Klasse wurde ein Event Listener angehängt: Dieser verursacht eine automatische Aktualisierung der Leinwand bei jedem relevanten Ereignis (Bild 7).

Benutzerinteraktionen und die Interaktionen von Elementen der Bühne miteinander lassen sich in EaseJS mit Leichtigkeit umsetzen. EaselJS unterstützt zwar WebGL, doch die Vorteile dieser Technologie kommen der Applikation nicht automatisch zugute.

Moderne Webbrowser können ihre Canvas-Berechnungen größtenteils bereits ohnehin auf die GPU auslagern. Durch den Einsatz von WebGL in EaselJS lassen sich dennoch Performancevorteile vom 6-Fachen bis zum 50-Fachen gegenüber 2D-Canvas erzielen. Hierbei ist es allerdings erforderlich, auf den Einsatz bestimmter Inhaltstypen zu verzichten und erweiterte Klassen zu nutzen. So kommt anstelle der Stage-Klasse zum Erzeugen einer gewöhnlichen Bühne die *SpriteStage*-Klasse zum Einsatz; anstelle gewöhnlicher Container arbeiten Sie mit der *SpriteContainer*-Klasse. *SpriteStage* beschränkt die *addChild()*-Methode auf Objekte, die sich mit WebGL leicht vertragen. Auch die *SpriteContainer*-Klasse limitiert die unterstützten Inhaltstypen, um das Rendering zu optimieren. Das DOM-Element funktioniert im Übrigen problemlos, da es nicht gerendert werden muss.

WebGL in EaselJS kann reibungslos auf den Rendering-Kontext Context2D einer gewöhnlichen Canvas zurückfallen, falls die WebGL-Unterstützung in dem jeweiligen Browser nicht verfügbar sein sollte. Einige Features, darunter Masken und Filter, lassen sich im WebGL-Modus von EaselJS allerdings nicht nutzen. Mit Hilfe einer Abfrage wie

```
if (mySpriteStage.isWebGL) {
  // WebGL-optimierter Code
```

▶

WebGL-Performance

Wie man Problemen mit der WebGL-Performance auf die Spur kommen kann: Falls Ihre WebGL-Animation keine zufriedenstellende Leistung erreicht, könnte es schlicht an der Grafikhardware liegen. Webbrowser schalten ihre WebGL-Fähigkeiten schon mal aus Stabilitätsgründen ab. Ein Dokument mit Informationen über Grafikhardware, die sich auf den sogenannten schwarzen Listen (black lists) der führenden Browserhersteller befindet, führt das Khronos-Konsortium auf.
www.khronos.org/webgl/wiki/BlacklistsAndWhitelists

```
  } else {
  // 2D-Canvas optimierter Code
}
```

können Sie die Verfügbarkeit von WebGL in Erfahrung bringen und zwei Varianten Ihrer Applikation bereitstellen: eine für den 2D-Kontext einer gewöhnlichen Canvas (der im Browser nativ gerendert wird) und eine andere Edition Ihres EaselJS-Codes für die hardwarebeschleunigte WebGL-Ausgabe in das *<canvas>*-Element hinein.

Der WebGL- und der Context2D-Rendering-Kontext von EaselJS lassen sich im Übrigen überlagern. Maus- und Touch-Interaktionen sowie Assets können hierbei zwischen den beiden Ebenen der Applikation reibungslos übergeben werden, um die Vorteile des jeweils am besten geeigneten Renderers zu nutzen. Eine wesentliche Einschränkung des WebGL-Renderers von EaselJS besteht darin, dass er 3D in WebGL leider nicht abdeckt.

Ein Oldsmobile: HelloRacer, ein interaktives 3D-Modell eines Formel-1-Boliden von HelloEnjoy, entstand unter Verwendung von ThreeJS noch zu den Anfangszeiten der Grafik-Revolution (Bild 8)

Three.js

Zu den beliebtesten JavaScript-Bibliotheken für anspruchsvolle Grafikausgaben unter Verwendung von WebGL zählt zweifelsohne Three.js. Three.js zählt nebenbei auch zu den Veteranen, die sich einen festen Platz unter den führenden 3D-Frameworks redlich verdient haben (Bild 8).

Ursprünglich in ActionScript geschrieben, wurde Three.js später in JavaScript umgesetzt. Diese leichtgewichtige 3D-Library kann interaktive, dreidimensionale Welten mit nur wenigen Zeilen Code ins Leben rufen und projektiert sie nativ oder mittels WebGL auf eine Canvas. Das Framework verfügt über eine modulare Rendering-Schnittstelle, die zusätzlich zu Canvas und WebGL auch noch SVG und CSS3D beherrscht.

Zu den Stärken von Three.js zählt eine umfassende Sammlung von Materialien, Lichtquellen und Geometriebibliotheken. Integrierte Unterstützung für 3D-Dateiformate aus Blender, Maya, Cinema4D und anderen Applikationen rundet das bemerkenswerte Leistungsspektrum ab. Three.js kommt auf ungefähr der Hälfte aller WebGL-fähigen Webseiten zum Einsatz.

Eine 3D-Animation in ThreeJS benötigt mindestens eine Szene, eine Kamera und einen Renderer sowie ein oder mehrere Objekte.

Unterstützt werden drei Typen von Kameras: PerspectiveCamera, OrthographicCamera und CubeCamera. Sie unterscheiden sich durch die verfügbaren Parameter. Die PerspectiveCamera-Klasse bietet einen bestimmten Blickwinkel auf eine 3D-Szene unter Berücksichtigung von Abständen zwischen Objekten der virtuellen Welt.

Die OrthographicCamera-Klasse ermöglicht eine parallele Projektion in 2D-Welten (ohne die Berücksichtigung der Perspektive). Die CubeCamera-Klasse ermöglicht die Handhabung von Spiegelungen der Umgebung auf der Oberfläche von Objekten (sie erzeugt hierzu jeweils sechs Kameras, um alle Richtungen abzudecken). Den Code einer Beispielanimation illustriert Listing 1.

Ein Ersatz für Flash/Silverlight: Eine reine 2D-Animation entsteht aus reinem Code in CreateJS und rendert im Browser in einem Canvas-Element ohne WebGL (Bild 7)

Listing 1: Eine 3D-Animation in Three.js

```
<!DOCTYPE html>
<html>
<head>
  <meta charset=utf-8>
  <title>Three.js-Test</title>
  <style>
    body { margin: 0; }
    canvas { width: 100%; height: 100% }
  </style>
</head>
<body>
  <script src="js/three.min.js"></script>
  <script>
    var scene = new THREE.Scene();
    var camera = new THREE.PerspectiveCamera(75,
    window.innerWidth / window.innerHeight, 0.1,
    1000);
    camera.position.z = 2;
    var renderer = new THREE.WebGLRenderer
    ({ antialias: true });
    renderer.setSize(window.innerWidth,
    window.innerHeight);
    document.body.appendChild(renderer.domElement);
    var geometry = new THREE.BoxGeometry(1, 1, 1);
    var material = new THREE.MeshPhongMaterial
    ({ color: 0x156289,
      emissive: 0x072534,
      side: THREE.DoubleSide,
      shading: THREE.FlatShading });
    var cube = new THREE.Mesh(geometry, material);
    scene.add(cube);
    var ambientLight = new THREE.AmbientLight(
    0x000000 );
      scene.add( ambientLight );

    var lights = [];
    lights[0] = new THREE.PointLight( 0xffffff, 1, 0 );
    lights[1] = new THREE.PointLight( 0xffffff, 1, 0 );
    lights[2] = new THREE.PointLight( 0xffffff, 1, 0 );

    lights[0].position.set( 0, 200, 0 );
    lights[1].position.set( 100, 200, 100 );
    lights[2].position.set( -100, -200, -100 );

    scene.add( lights[0] );
    scene.add( lights[1] );
    scene.add( lights[2] );

    function render() {
      requestAnimationFrame( render );
      cube.rotation.x += 0.01;
      cube.rotation.y += 0.01;
      renderer.render( scene, camera );
    }
    render();
  </script>
</body>
</html>
```

Der Verweis auf die Three.js-Bibliothek und das Skript, aus dem das *<canvas>*-Element herausgegeben wird, werden in den *<body>*-Abschnitt der HTML5-Datei eingefügt. Zur Initialisierung der Szene genügt:

```
var scene = new THREE.Scene();
```

Im Beispiel kommt eine Perspective-Camera zum Einsatz. Sie verfügt über die Parameter Sichtfeld, Seitenverhältnis sowie nahe und ferne Clipping-Ebenen:

```
var camera = new THREE.PerspectiveCamera(75,
window.innerWidth / window.innerHeight, 0.1, 1000);
```

Der Parameter *camera.position* definiert die Platzierung der perspektivischen Kamera in der Tiefenachse (z) und bestimmt somit indirekt den Skalierungsfaktor der Beispielanimation.

Bei der Gestaltung von 3D-Welten kommt der leistungsstarke WebGLRenderer von Three.js zum Einsatz:

```
var renderer = new THREE.WebGLRenderer
({ antialias: true });
```

Wurde eine Instanz des Renderers erst einmal erzeugt, gilt es, die Dimensionen der Ausgabe festzulegen. Im Beispiel werden diese auf die Größe des Viewports gesetzt:

```
renderer.setSize(window.innerWidth, window.innerHeight);
```

Wer aus Performancegründen die Ausgabe der App vorerst mit verringerter Auflösung rendern lassen möchte, kann diese Zeile etwa wie folgt anpassen, um die Auflösung pro Achse zu halbieren:

```
renderer.setSize(window.innerWidth/2,
window.innerHeight/2, false);
```

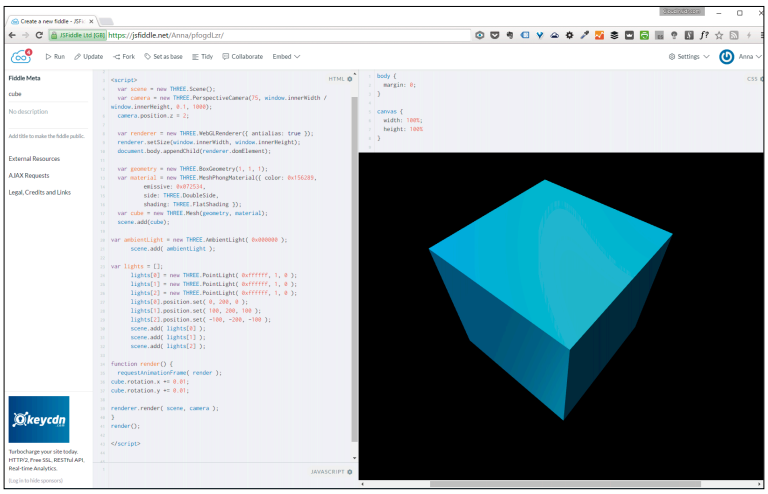
Dadurch hantiert der Renderer mit einem Viertel der Gesamtauflösung der Ausgabe. Schließlich wird das Renderer-Objekt an das passende DOM-Element angehängt:

```
document.body.appendChild(renderer.domElement);
```

Ein Würfel entsteht unter Verwendung der Geometrieklasse *BoxGeometry* mit den obligatorischen Parametern Breite, Höhe und Tiefe: ▶

```
var geometry = new THREE.BoxGeometry( 1, 1,
1 );
var material = new THREE.MeshBasicMaterial( {
color: 0x00ff00 } );
var cube = new THREE.Mesh
( geometry, material );
scene.add( cube );
```

Doch Vorsicht: Nicht alle Materialien unterstützen alle Effekte. Auf *MeshBasicMaterial* etwa lassen sich keine Lichteffekte anwenden. Für wen das ein Problem darstellt, der sollte das Material wechseln (zum Beispiel MeshLambertMaterial oder MeshPhongMaterial mit zusätzlichen Parametern).



Glatt gelaufen: die Beispielanimation in ThreeJS auf der Basis von WebGL (Bild 9)

Lichtquellen und andere Effekte

Three.js unterstützt insgesamt sechs Arten von Lichtquellen, die eine Vielzahl von Effekten erzeugen können. Um der Szene dezentes Umgebungslicht hinzuzufügen, genügt:

```
var ambientLight = new THREE.AmbientLight( 0x000000 );
scene.add( ambientLight );
```

Gerichtete Lichtquellen entstehen so:

```
var lights = [];
  lights[0] = new THREE.PointLight( 0xffffff, 1, 0 );
  lights[1] = new THREE.PointLight( 0xffffff, 1, 0 );
  lights[2] = new THREE.PointLight( 0xffffff, 1, 0 );
  lights[0].position.set( 0, 200, 0 );
  lights[1].position.set( 100, 200, 100 );
  lights[2].position.set( -100, -200, -100 );
  scene.add( lights[0] );
  scene.add( lights[1] );
  scene.add( lights[2] );
```

Die Three.js-Bibliothek übernimmt die Berechnungen dieser Effekte unter Berücksichtigung der gewählten Materialien und des Ablaufs der Animation im 3D-Raum (Bild 9) – Aufgaben, die sich ohne ein geeignetes Framework extrem spannend gestalten würden.

Mit Hilfe von Three.js lassen sich realitätsnahe 3D-Modelle und atemberaubende 3D-Welten gestalten und mit flüssiger Leichtigkeit und glaubwürdiger Detailtreue im Browser darstellen. Die Einarbeitung in Three.js zählt nicht zu den einfachsten Aufgaben. Die Three.js-Entwicklergemeinde leistet tatkräftige technische Unterstützung auf StackOverflow.

BabylonJS

Bei BabylonJS handelt es sich um eine der leistungsstärksten und innovativsten quelloffenen 3D-Engines auf der Basis von WebGL und JavaScript/TypeScript. Das Framework entstand ursprünglich mit Unterstützung von Microsoft als eine Spiele-Engine für Silverlight und wurde mit der Zeit zum Herausforderer für den Platzhirsch Three.js. Mit Hilfe von BabylonJS können Sie komplexe geometrische Formen animieren, um anspruchsvolle Simulationen und Spiele zu gestalten. David Catuhe, der Urheber der ersten Iteration der Engine, hatte aus seinen Erfahrungen mit DirectX, OpenGL und Silverlight reichlich Inspiration geschöpft und bei führenden 3D-Software-Tools Anleihen gemacht. Das Resultat ist eine bemerkenswerte Vielfalt an Features, die (auch) einer professionellen 3D-Compositing-Suite gut zu Gesicht stünden: Szenengraphen mit Lichtquellen, bewegliche Kameras, eine umfassende Sammlung von Materialien und Drahtgittermodellen (Bild 10), eine Kollisionssimulations-Engine auf der Basis von CannonJS, eine Physikeffekt-Engine mit Partikelsystemen, Spezialeffekte wie Nebel, Blendenflecke und Postproduktionsweichzeichner, eine Audio-Engine und zahlreichen Optimierungsmöglichkeiten für die verschiedenen Eigenschaften der Objekte und ihr Verhalten auf der Bühne. Mit erweiterten Fähigkeiten zur automatischer Kollisionserkennung und intelligentem Antialiasing glänzt Babylon vor allem bei Spielen, meistert aber auch virtuelle Realität mit Bravour.

BabylonJS-Szenen lassen sich aus Dateien in den Formaten OBJ, FBX und MXB wiederherstellen und für Blender, Cheetah3d und Cinema 3ds max exportieren.

Die Umsetzung einer BabylonJS-Animation illustriert Listing 2. Die Shader sind in BabylonJS entweder in *<script>*-Tags eingeschlossen und mit dem MIME-Typ *application/vertexShader* beziehungsweise *application/fragmentShader* gekennzeichnet. Alternativ lassen sich die Shader aus separaten FX-Dateien beziehen. Das eigentliche JavaScript, der die Szene steuert, illustriert Listing 3.

Der Einsatz von *BABYLON.ShaderMaterial* erlaubt es, sich um die aufwendige Kompilierung, das Linking und die sonstige Handhabung der Shader herumzuarbeiten. Eine ganze Menge Programmierarbeit entfällt dadurch ganz elegant. Beim Einsatz des Materials muss man allerdings entweder das DOM-Element spezifizieren, in dem sich die Shader befinden, oder aber diejenigen FX-Dateien, welche die Shader beinhalten, nach den Namenskonventionen des Frameworks benennen *(shadername.vertex.fx* und *shadername.fragment.fx)*. In diesem Fall müssen Sie allerdings die Shader unter An-

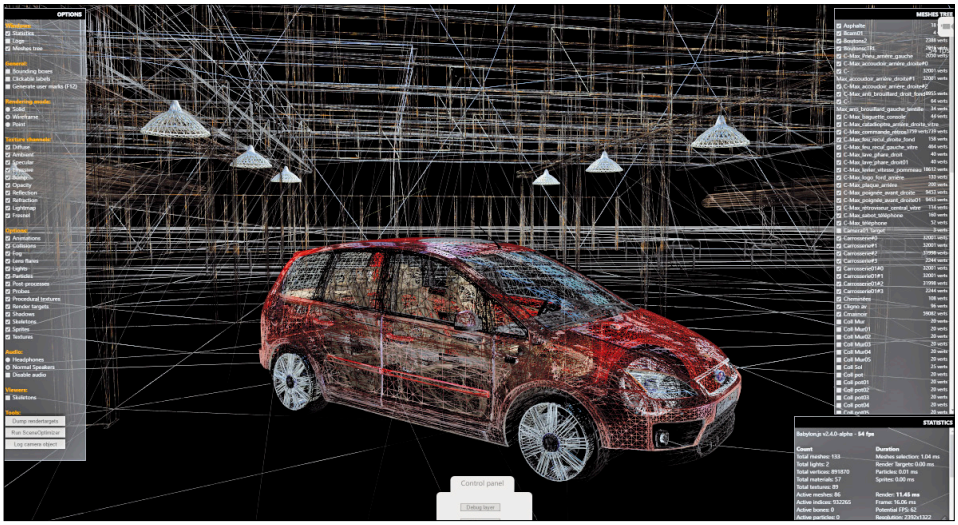
gabe des Pfades zu den Shader-Dateien referenzieren und die Namen der Attribute und Uniform-Variablen an den Konstruktor als Optionen übergeben:

```
var neuesMaterial = new BABYLON.
ShaderMaterial("materialName", scene, "./ shadername",
{
  attributes: ["position", "uv"],
  uniforms: ["SpielWeltProjektion"] });
```

Wenn Sie die Uniform-Variablen korrekt deklarieren, können Sie nämlich die Funktionen *setTexture, setFloat, setFloats, setColor3, setColor4, setVector2, setVector3, setVector4* und *setMatrix* nutzen, um die benötigten Werte zu setzen. Der *BABYLON.ShaderMaterial*-Konstruktor wird dann die benötigte Matrix jedes Mal auch automatisch für Sie berechnen.

BabylonJS war traditionell sehr codezentrisch; es fehlten einfach jegliche Tools, die die Nutzung der umfassenden Fähigkeiten dieser Bibliothek erleichtern könnten. Neuerdings scheint sich dieser Missstand zu ändern. Unter der Adresse www.babylonjs-playground.com finden Sie einen interaktiven Editor von BabylonJS-Szenen. Zwar ist dieser Dienst noch in der Entwicklung, doch es zeichnen sich bereits nützliche Features ab.

Der umfassende Funktionsumfang von BabylonJS ist vielen (vor allen mobilen) Entwicklern schlicht zu viel des Guten. Eine leichtgewichtige Edition von BabylonJS für Haxe namens BabylonHx ist inzwischen unter der Adresse http://babylonhxgamestudiohx.com erhältlich.



Der Durchblick: die Drahtgitter-Ansicht eines 3D-Modells aus BabylonJS im Webbrowser (Bild 10)

BabylonHx

BabylonHx ist eine ultra-portable, Cross-Plattform-fähige und leichtgewichtige 3D-Engine auf der Basis von BabylonJS. Bei Haxe handelt es sich um ein quelloffenes Toolkit auf der Basis der gleichnamigen Programmiersprache für iOS, Android, HTML5, Neko, Node.js, C++, C#, PHP und Java, das unter der Patronage der Haxe Foundation entstanden ist (http://haxe.org). Bei Neko handet es sich um eine High-Level-Programmiersprache von Nicolas Cannasse mit dynamischer Typüberprüfung, einem Compiler und einer VM. Nekos VM bildet eine vereinheitlichte Runtime für verschiedene andere Sprachen und lässt sich selbst als eine Skriptsprache einbinden. Der Compiler konvertiert Neko-Quellcode in Bytecode, der sich in der Neko-VM ausführen lässt. Haxe kompiliert den eigenen Quellcode in andere Sprachen.

BabylonHx ermöglicht somit die Entwicklung und Einbindung von 3D-Welten in native mobile Apps (ausgehend von einer einheitlichen Code-Basis) auf einer Vielzahl von Plattformen. BabylonHx arbeitet Hand in Hand mit OpenFL, einer quelloffenen Entwicklungsplattform, und Snow, einem Low-Level-Entwicklungsframework für Haxe.

OpenFL kann unter anderem für Windows, Mac, Linux, iOS, Android, BlackBerry, Firefox OS, Tizen, Flash und HTML5 ausgeben und deckt somit die überwältigende Mehrheit internetfähiger Endgeräte ab. Snow (derzeit noch als Alpha-Version vorliegend) lässt sich mit zahlreichen Code-Editoren, darunter Sublime Text (Mac/Windows/Linux), Atom (ebenfalls Mac/Windows/Linux) und FlashDevelop (derzeit nur unter Windows) nutzen und bietet die Ausgabe für OS X, Windows, Linux, iOS und Android sowie WebGL an.

BabylonHx zeigt eine bemerkenswerte Flexibilität, wenn es darum geht, für das *<canvas>*-Element in HTML5 auszugeben. BabylonHx sieht hierzu drei Workflows vor:

- standalone: durch Erstellen eines Haxe-JS-Projekts und die Übergabe des *purejs*-Parameters an den Haxe-Compiler,
- in einem Lime/OpenFl- oder Snow-Projekt und die Ausgabe für JS/HTML5,
- nach dem Vorbild eines klassischen BabylonJS-Workflows: durch das Kompilieren der BabylonHx-Engine in eine JavaScript-Datei.

Um in BabylonHx zu entwickeln, müssen Sie allerdings einige Tools einrichten, allen voran den Haxe-Compiler und Neko VM. Darüber hinaus können Sie eine beliebige Kombination aus Lime/Openfl, NME und Snow einspannen. Nach der Konfiguration von Lime müssen Sie für jede avisierte Plattform – mit der Ausnahme von HTML5/Canvas/WebGL – die zugehörigen Entwickler-Tools installieren.

RGraph

RGraph ist eine quelloffene JavaScript-Engine für die Gestaltung von Graphen und Diagrammen auf der Basis von *<canvas>* in HTML5 mit der Fähigkeit, sich dynamisch zu aktualisieren. RGraph-basierte Diagramme rendern bis zu zehnfach schneller als vergleichbare Grafiken in Flash oder ▶

SVG. Die Bibliothek unterstützt circa 60 Typen von Visualisierungen, darunter Balken-, Linien- und Kreisdiagramme, interaktive Kraftstoffanzeigen, diverse Fortschrittsbalken, Entfernungsmesser, Thermometer, Radar-Charts, Gantt-Diagramme und dergleichen mehr. Das Design ist jedoch sehr einfach gehalten und besticht auch nicht durch überwältigende Eleganz (Bild 11).

Für die neue Version der Bibliothek, die in Kürze verfügbar sein wird, hat der Entwickler eine Reihe von Verbesserungen angekündigt. Zu den Neuerungen zählt unter anderem auch barrierefreier Text, ein Feature, das die Bibliothek für den Einsatz durch Behörden und öffentliche Einrichtungen nutzbar macht, die eine hohe Barrierefreiheit ihrer Webapplikationen anstreben.

Die RGraph-Bibliothek wird im *<head>*-Abschnitt des HTML-Dokuments wie folgt eingebunden:

```
<script src="RGraph.common.core.js"></script>
<script src="RGraph.bar.js"></script>
```

Im *<body>*-Bereich wird ein *<canvas>*-Element benötigt:

```
<canvas id="cvs" width="700" height="250">
  Ihr Browser kann die Canvas nicht rendern.
</canvas>
```

Zu guter Letzt wird die Grafik gezeichnet (im Beispiel ein Balkendiagramm mit barrierefreiem Text):

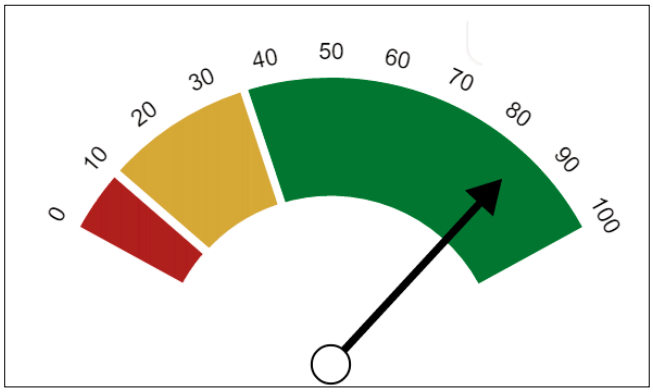
```
<script>
  var bar = new RGraph.Bar({
    id: 'cvs',
    data: [11,16,9,7,5,22,24],
    options: {
      labels: ['Montag','Dienstag','Mittwoch',
      'Donnerstag','Freitag','Samstag','Sonntag'],
      shadow: false,
      colors: ['red'],
      strokestyle: 'rgba(0,0,0,0)',
      textColor: '#666'
    }
  }).draw();
</script>
```

Die Anpassung der Diagramme ist also denkbar einfach. Das Framework bietet zudem Klassen zum Importieren von Daten aus CSV-Dateien und Google Sheets. Eine Datenbankschnittstelle gibt es zwar nicht, dafür lässt sich das Ziel mit einem Workaround bewerkstelligen.

Zu den Vorzügen von RGraph zählt die Fähigkeit der Bibliothek, numerische Werte zu animieren und einige Diagram-

Listing 2: Ein animierter 3D-Ball in BabylonJS

```
<!DOCTYPE html>
<html>
<head>
  <title>Babylon.js</title>
  <script src="http://www.babylonjs.com/hand.
  minified-1.2.js"></script>
  <script src="http://www.babylonjs.com/babylon.js">
  </script>
  <script type="application/vertexShader"
  id="vertexShaderCode">
    precision highp float;
    attribute vec3 position;
    attribute vec2 uv;
    uniform mat4 SpielWeltProjektion;
    varying vec2 vUV;

    void main(void) {
      gl_Position = SpielWeltProjektion *
      vec4(position, 1.0);
      vUV = uv;
    }
  </script>
  <script type="application/fragmentShader"
  id="fragmentShaderCode">
    precision highp float;
    varying vec2 vUV;
    uniform sampler2D textureSampler;
    void main(void) {
      gl_FragColor = texture2D(textureSampler, vUV);
    }
  </script>
  <script src="index.js"></script>
  <style>
    html, body {
      width: 100%;
      height: 100%;
      padding: 0;
      margin: 0;
      overflow: hidden;
      margin: 0px;
      overflow: hidden;
    }
    #renderCanvas {
      width: 100%;
      height: 100%;
      touch-action: none;
      -ms-touch-action: none;
    }
  </style>
</head>
<body>
  <canvas id="renderCanvas"></canvas>
</body>
</html>
```



Minimalistisch: RGraph verzichtet auf effektvolles Design zugunsten von unaufdringlicher Simplizität (Bild 11)

me in 3D zu zeichnen. Die fehlende Unterstützung für WebGL führt allerdings dazu, dass sich Diagramme in 3D aufgrund der schwachen Renderingleistung relativ schlecht mit Schatten vertragen.

Die Lizenzbedingungen von RGraph erlauben eine kostenfreie Nutzung auch für gewerbliche Zwecke unter der GPL-Lizenz. Zu den Lizenzbedingungen zählt allerdings ein Backlink auf Rgraph.net. Wem die Bedingungen der GPL-Lizenz zu restriktiv erscheinen, der kann die Bibliothek unter einer kostenpflichtigen, kommerziellen Lizenz oder unter einer ebenfalls kostenpflichtigen OEM-Lizenz einsetzen.

ImpactJS

Bei Impact handelt es sich um eine beliebte JavaScript-Spiele-Engine des deutschen Entwicklers Dominic Szablewski aus dem Hessischen Büttelborn zur Entwicklung von HTML5-Spielwelten für Desktops und Mobilgeräte auf der Basis von Canvas in 2D (Bild 12). Impact glänzt unter anderem durch eine gelungene Handhabung von Kollisionen. Die Bibliothek beinhaltet Ejecta, eine quelloffene Implementierung von JavaScript, Canvas und HTML5-Audio für Spiele und Animationen auf iOS und tvOS.

Ejecta führt den JavaScript-Code in einer JavaScript VM (JavaScriptCore) aus. Ejecta erleichtert die Portierung eines Impact-Spiels auf iOS und integriert sich im Übrigen unter anderem mit ThreeJS. Die Canvas 2D- und WebGL APIs wurden in nativem Code unter Verwendung von OpenGL umgesetzt. Die Implementierung von Audio basiert auf OpenAL.

Ejecta benötigt iOS 8.0 oder neuer mit Unterstützung für den ARMv7- oder den ARM64-Befehlssatz (iPhone 4S und neuer und iPads ab der zweiten Generation). Es unterstützt alle Retina-Auflösungen und alle Display-Größen. Zum Lieferumfang von Impact gehört Weltmeister Level Editor, eine browserbasierte Applikation für die grafische Bearbeitung von Szenen.

Goo Engine/Goo Create

Bei Goo handelt es sich um mehr als eine JavaScript-Bibliothek: Goo ist eine Plattform für die visuelle Entwicklung von 2D- und 3D-Welten auf der Basis der quelloffenen Goo Engine (GooJS) in einem GUI-Editor namens Goo Create.

Listing 3: Ein animierter 3D-Ball in BabylonJS

```
document.addEventListener("DOMContentLoaded",
startGame, false);
function startGame() {
  if (BABYLON.Engine.isSupported()) {
    var canvas = document.
    getElementById("renderCanvas");
    var engine = new BABYLON.Engine(canvas, false);
    var scene = new BABYLON.Scene(engine);
    var camera = new BABYLON.ArcRotateCamera(
    "Camera", 0, Math.PI / 2, 10, BABYLON.Vector3.
    Zero(), scene);
    camera.attachControl(canvas);
    var sphere = BABYLON.Mesh.CreateSphere
    ("Sphere", 16, 5, scene);
    var amigaMaterial = new BABYLON.
    ShaderMaterial("amiga", scene, {
      vertexElement: "vertexShaderCode",
      fragmentElement: "fragmentShaderCode",
    },
    {
      attributes: ["position", "uv"],
      uniforms: ["SpielWeltProjektion"]
    });
    amigaMaterial.setTexture("textureSampler",
    new BABYLON.Texture("amiga.jpg", scene));

    sphere.material = amigaMaterial;
    engine.runRenderLoop(function () {
      sphere.rotation.y += 0.05;
      scene.render();
    });
  }
};
```

Goo Create läuft in einem Webbrowser und bietet hier das Leistungsspektrum einer 3D-Animations- und Compositingsoftware. In Goo Create lassen sich 3D-Objekte mit der Maus zusammenstellen und grafisch animieren, Kameras erstellen und bewegen et cetera. Die Resultate sprechen für sich.

Die Idee von Goo Create mag sehr lobenswert sein, doch die Umsetzung hat zwei grundlegende Schwachstellen. Durch die unzureichende Erweiterbarkeit der Goo Engine und das völlige Fehlen kostenfreier Werkzeuge besteht die Gefahr, dass die Plattform im Lauf der Zeit an Zugkraft verliert, weil sich nur relativ wenige Entwickler daran beteiligen. Eine aktive Beteiligung fällt einem im Übrigen angesichts der nutzungsfeindlichen Preisstruktur für Goo Create nicht ganz so leicht. Um nicht auf die freundliche Hilfe in den Foren angewiesen zu sein, sondern in den Genuss des Premium-Supports zu kommen, benötigt man ein monatliches Abonnement in der Ausbaustufe Pro oder Custom (ab 49 US-Dollar pro Benutzer pro Monat) und muss dann auch noch ein Support-Abonnement erwerben, das – völlig nutzungsunabhängig ▶

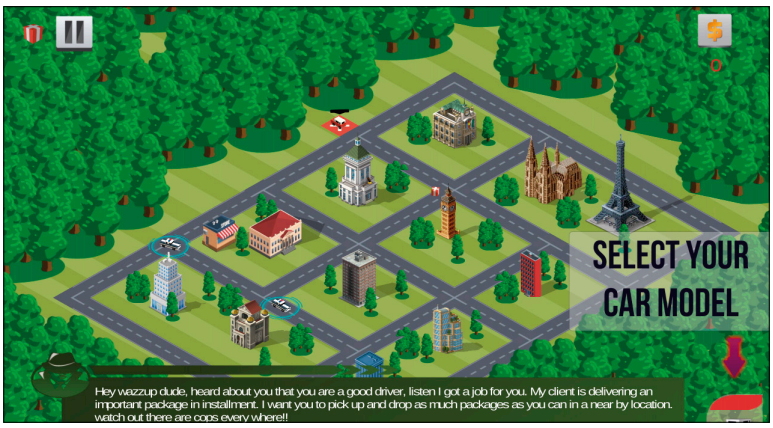
– ebenfalls mit einem monatlichen Preis ab 100 US-Dollar zu Buche schlägt.

Dafür lassen sich in Goo Create Szenen mit grafischen Werkzeugen – ohne die direkte Bearbeitung des Codes – erstellen und animieren.

Berechnungen von Naturphänomenen

Für die Berechnungen von Naturphänomenen wie den Ablauf von Kollisionen zwischen Objekten oder die Wirkung von Gravitationskräften zeichnet in Goo eine externe quelloffene Bibliothek namens CannonJS verantwortlich.

Diese weniger als 100 KByte große Datei (angelehnt an die Funktionsweise von Three.js mit ammo.js) liefert eine glaubwürdige Umsetzung von Interaktionen zwischen Objekten einer Szene. So lassen sich ganze Spielwelten als virtuelle Realität umsetzen. CannonJS stellt Ihnen ein leistungsstarkes API bereit, das dafür sorgt, dass sich die Bibliothek mit jedem beliebigen Renderer auf Basis von Canvas, WebGL oder CSS3 nutzen lässt.



Abgeholt: Pick'N Drop, ein mobiles 2D-Spiel für die Canvas (Bild 12)

Goo Create bringt eine Sammlung von Skripts mit, welche die Bewegung von Elementen einer Szene kontrollieren können. Doch erst durch den geschickten Einsatz eigener, maßgeschneiderter Skripts kommt das volle Leistungsspektrum der Goo-Plattform in WebGL zum Vorschein.

Hierzu können Sie in dem grafischen Editor Goo Create an die einzelnen Objekte Ihrer Szene jeweils ein sogenanntes ScriptComponent anhängen und das zugehörige Skript im integrierten Skripteditor bearbeiten.

Fazit

Webgrafik-Frameworks, die auf das in HTML5 standardisierte *<canvas>*-Element rendern können, gibt es mittlerweile wie Sand am Meer. Die Schwergewichte unterstützen allesamt hardwarebeschleunigtes Rendering von 3D-Welten mittels WebGL.

Die wichtigsten Alleinstellungsmerkmale dieser Frameworks liegen in der Handhabung von Kollisionen der Szenenelemente untereinander und der Abarbeitung von Benutzerinteraktionen mit der Bühne.

Da die technischen Herausforderungen rund um die Bereitstellung von High-End-Grafik ohne Plug-ins in den aktuellen Webbrowsern bereits gelöst sind, dürfte sich künftig die Musik bei den grafischen Werkzeugen abspielen. Vulkan, der kürzlich angekündigte und überaus würdige Nachfolger von OpenGL, dürfte in nicht ganz so ferner Zukunft für einen weiteren Leistungsschub bei der GPU-Performance sorgen und die Kreativität der Spieleentwickler weiter herausfordern. Wer bereits heute auf ein leistungsstarkes Canvas-Framework setzt, darf sich auf eine spannende Zukunft freuen. ■

Links zum Thema

- ACE
  https://ace.c9.io/#nav=about
- Sylvester
  https://github.com/jcoglan/sylvester
- OpenFL
  www.openfl.org
- Download von Haxe
  http://haxe.org/download
- Neko
  http://nekovm.org
- Snow für Haxe
  http://underscorediscovery.github.io/snow
- CannonJS
  www.cannonjs.org
- Ammo.js
  https://github.com/kripken/ammo.js
- CreateJS
  www.createjs.com
- Three.js
  http://threejs.org
- BabylonJS
  www.babylonjs.com
- BabylonHx
  http://babylonhx.gamestudiohx.com
- RGraph
  www.rgraph.net
- ImpactJS
  http://impactjs.com
- Goo Engine/Goo Create
  https://github.com/GooTechnologies/goojs

Filipe Martins und Anna Kobylinska sind international anerkannte IT-Berater mit Schwerpunkt auf Cloud-Lösungen. **@D1gitalPro** und **@D1gitalInfo**

DDC

.NET Developer Conference 2016

05. – 07. Dezember 2016 + Special am 08.12.2016
Köln, pullman Cologne

DevSessions, Konferenz + Workshops

Tools + Technologien:
Softwarequalität, Frontend, Core, Any App
dotnet-developer-conference.de

SMART DATA Developer Conference

Big Data & Smart Analytics

06. – 07. Dezember 2016
Köln, pullman Cologne

Konferenz + Workshop

Themen: Datenqualität, Visualisierung, Analyse, Batch & Stream, Processing, Tools & Frameworks
smart-data-developer.de

RENDERING-PERFORMANCE VON WEBSEITEN UND WEB-APPS

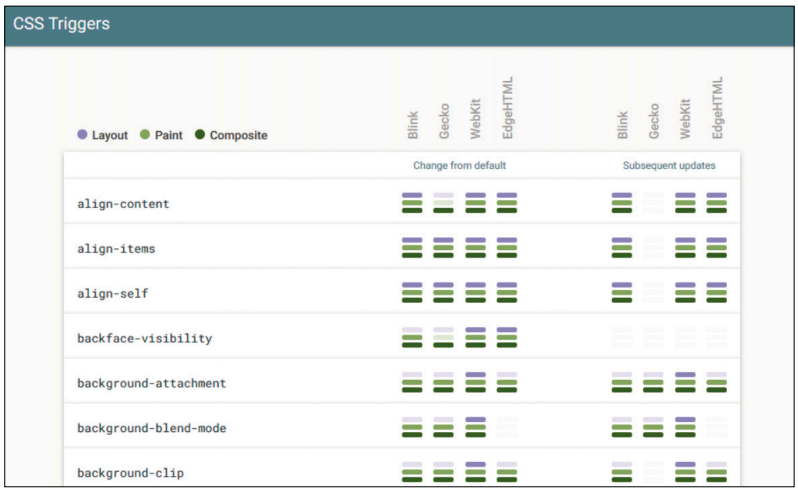
Ruckelfrei

Interaktionen auf Webseiten sollten flüssig und geschmeidig ablaufen.

Bei dem Begriff Performance denkt man meist zuerst an die Geschwindigkeit, mit der die Daten für eine Webseite übertragen werden. Bei der Rendering-Performance geht es hingegen darum, wie die Daten dargestellt werden, und dass Komponenten schnell auf Aktionen durch den Benutzer reagieren. Animationen müssen flüssig ablaufen und dürfen nicht ruckeln – nur dann empfinden die Benutzer die Interaktionen als positiv. Damit das klappt, müssen die Daten dem Browser möglichst effektiv zur Verfügung gestellt werden und Interaktionen/Animationen richtig konzipiert sein.

Um zu verstehen, wie sich die Rendering-Performance beeinflussen lässt, sollte man sich ansehen, wie ein Browser eine Seite darstellt. Entscheidend ist erst einmal die Framerate: Die meisten Geräte erneuern die Bildschirmdarstellung 60-mal in der Sekunde. Höhere Frameraten bringen nichts, weil sie vom Benutzer nicht wahrgenommen werden. Bei Interaktionen oder beim Scrollen durch den Benutzer muss der Browser der Aktualisierungsrate des Geräts entsprechen und für jede dieser Bildaktualisierungen ein neues Bild zur Verfügung stellen.

Für eine einzelne Aktion stehen damit 16 ms zur Verfügung. Von diesen 16 ms muss man jedoch noch etwas abziehen, da die Browser ja auch noch etwas anderes zu tun haben – sodass als Richtwert 10 ms gelten. Einzelne Operationen sollten also nie länger als 10 ms dauern.



Welche Bereiche sind bei der Änderung einzelner CSS-Eigenschaften involviert? CSS Triggers liefert die Antwort (Bild 2)

Pixel-Pipeline

Kommen wir zu der Frage, wie ein Browser eine Seite darstellt und welche Bereiche hierbei involviert sind (Bild 1). Wenn es um Interaktionen oder Animationen geht, beginnt alles damit, dass über JavaScript oder auch über CSS eine Änderung angestoßen wird.

Den nächsten Bereich nennt Paul Lewis bei Google Developers Style. Dabei geht es darum, dass die Elemente ausgewählt werden müssen, die formatiert werden sollen. Der Browser muss herausfinden, für welche Elemente eine Formatierung gilt, und die endgültige Formatierung berechnen. Wenn Browser wissen, welche Elemente wie formatiert werden sollen, müssen die Ausmaße und die genaue Position der Elemente auf dem Bildschirm berechnet werden (Layout). Dabei kommt es häufig vor, dass Elternelemente die Position ihrer Nachfahren-Elemente beeinflussen.

Als Nächstes folgt das eigentliche Zeichnen der Elemente, Paint. Hier werden die Pixel gefüllt, Text, Farben, Bilder, Rahmen und Schatten erstellt. Meist findet das Zeichnen auf mehreren Ebenen statt. Diese Ebenen können Sie sich ähnlich vorstellen wie die Ebenen bei Photoshop oder einem anderen Grafikprogramm.

Der letzte Schritt besteht im Zusammenfügen der gezeichneten Ebenen. Dieser Vorgang wird Composite genannt.

JavaScript, Style, Layout, Paint, Composite sind die Schritte, die der Browser durchläuft. Aber es müssen nicht bei allen Operationen alle Bereiche involviert sein. Beispielsweise kann die Layout-Komponente fehlen, wenn bei einer Animation nur die Farben geändert werden. Am einfachsten sind Prozesse, bei denen weder Paint noch Layout durchgeführt wird, sondern nur Composite.

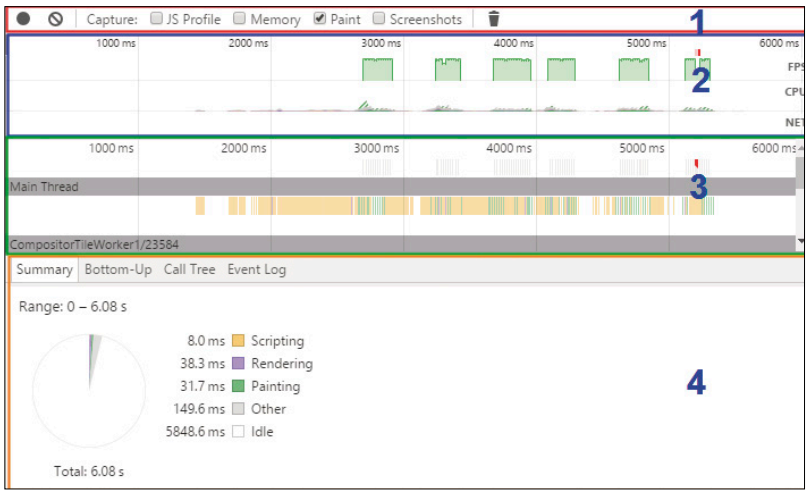
Einen Überblick darüber, welche Bereiche bei einzelnen CSS-Eigenschaften involviert sind, liefert CSS Triggers (Bild 2). Bevor wir uns das an einem Beispiel ansehen, kommen wir zu einem Tool, das bei der Untersuchung der Rendering-Performance hilft.

Die sogenannte Pixel Pipeline – die einzelnen Schritte zum Neuzeichnen (Bild 1)

Die aktuellen Browser bieten von Haus aus gute Werkzeuge zur Analyse der Performance. In Google Chrome ist es die Timeline. Diese können Sie, nachdem Sie die Entwickler-Tools aufgerufen haben, im oberen Bereich durch Klick auf *Timeline* öffnen.

Die Anzeige ist in verschiedene Bereiche unterteilt (Bild 3): Ganz oben ist der Steuerungsbereich (1). Hier befindet sich links der Kreis, über den Sie eine Aufzeichnung starten. Der durchgestrichene Kreis daneben löscht die Ausgaben. Dahinter kann man auswählen, was alles berücksichtigt werden soll, und über den Papierkorb eine Garbage Collection auslösen.

Darunter (2) befindet sich der Überblicksbereich mit einer Zusammenfassung der Performance, unterteilt in FPS, CPU und NET – dazu gleich mehr. Unterhalb des Überblicksbereichs ist der FlameChart mit dem Stacktrace (3) platziert. Noch etwas weiter unten finden Sie einen Bereich für Details (4) – je nachdem, was ausgewählt ist, sieht man hier weitere Informationen zu dem Zeitframe.



Die Timeline in den Entwickler-Tools von Chrome zur Untersuchung der Performance (Bild 3)

Damit im Überblicksbereich, der Informationen über FPS, CPU und NET bereitstellt, etwas zu sehen ist, muss erst einmal etwas aufgezeichnet sein. FPS steht für frames per second. Je höher die grüne Leiste, desto höher ist die Anzahl an Frames pro Sekunde. Wenn rote Spitzen zu sehen sind, ist das ein Indiz für ein mögliches Ruckeln. CPU-Ressourcen: Hier informiert das Flächendiagramm darüber, welche Art von Event welche CPU-Ressourcen verbraucht.

Bei NET geht es um das Netzwerkübertragung: Jede Linie repräsentiert dabei eine Ressource. Je länger der Balken, desto länger hat es gedauert, die Ressource zu laden. Der helle Teil des Balkens gibt die Zeitspanne bis zum Laden des ersten Bytes an. Der dunklere Bereich informiert über die Zeit zwischen dem ersten und letzten geladenen Byte.

Aufzeichnung starten

Sie können mit der Timeline sowohl das Laden der Seite als auch das Verhalten bei Interaktionen mitprotokollieren. Für ein Protokoll des Ladevorgangs müssen Sie bei geöffnetem Timeline-Tab die Seite neu laden. Daraufhin findet eine Aufzeichnung des Pageload statt und die Aufzeichnung endet automatisch, wenn die Seite vollständig geladen ist.

Wollen Sie hingegen das Verhalten bei bestimmten Interaktionen protokollieren, legen Sie Beginn und Ende per Klick auf den Aufzeichnungsbutton im Kontrollbereich oben links fest. Während der Aufzeichnung wird der Button rot. Die Aufzeichnung sollten Sie möglichst kurz halten, sich auf das Problem konzentrieren und unnötige Aktionen vermeiden. Zudem empfiehlt es sich, den Browsercache zu deaktivieren.

Sie können dann im Überblicksbereich einen Bereich der Aufzeichnung markieren, um ihn genauer zu betrachten. Damit wird der FlameChart automatisch vergrößert, um den entsprechenden Bereich wiederzugeben.

Im FlameChart lassen sich durch Klick auf ein Ereignis weitere Informationen anzeigen – und beispielsweise feststellen, wie lange die Ausführung eines Skripts gedauert hat. Bei der Aufzeichnung können Sie auch durch Klick auf *Screenshots* im Kontrollbereich Screenshots der einzelnen Zustände anfertigen lassen. Außerdem erlaubt es die Auswahl von *Paint,* die Zeichenvorgänge zu beobachten, und *JS Profile* hilft bei der Untersuchung des JavaScript-Codes.

Um verschiedene Versionen zu vergleichen, können Sie Timeline-Aufzeichnungen abspeichern: Dafür klicken Sie im Übersichtsbereich mit der rechten Maustaste und wählen den entsprechenden Befehl im Kontextmenü. Auf dieselbe Weise lässt sich eine bereits aufgezeichnete Timeline erneut für die Untersuchung laden.

Ähnlich wie die Timeline von Chrome funktioniert die Laufzeitanalyse in den Entwickler-Tools von Firefox, beziehungsweise der Punkt *Performance* in der Firefox Developer Edition.

Häufig sind es gerade die Interaktionen auf schwachbrüstigeren Geräten wie Smartphones, die ruckelnd ablaufen. Da hilft die Option des *Remote Debugging* für Android-Geräte, die über Chrome möglich ist. Hierfür muss das USB-Debugging im Android-Gerät aktiviert sein. Dann können Sie über die drei Punkte in der Menüleiste der Chrome-Entwickler-Tools unter *More Tools* den Punkt *Inspect Devices* aufrufen.

Wenn alles richtig konfiguriert und das Gerät verbunden ist, können Sie die Aktionen live auf dem Android-Gerät verfolgen und untersuchen. Eine vergleichbare Funktionalität für iOS-Geräte erlaubt der Webinspektor von Safari, oder Sie können dafür auch die Firefox-Developer-Edition nutzen.

JavaScript-Animationen

Für Animationen können Sie heute JavasScript oder CSS nutzen. In JavaScript selbst stehen die klassischen Methoden *setTimeout()* oder *setInteral()* für zeitgesteuerte Abläufe zur Verfügung. Das Problem an diesen Methoden ist jedoch, dass ihre Aktionen eventuell zu einem ungünstigen Zeitpunkt ausgeführt werden, beispielsweise wenn der Browser nicht beim Zeichnen ist. Das verbraucht Prozessorleistung und eventuell Akkuladung. Zudem kann es passieren, dass ▶

durch den ungünstigen Zeitpunkt ein Frame ausgelassen wird und gerade dadurch die Darstellung ruckelt.

Besser ist es, auf *requestAnimationFrame()* zurückzugreifen. Hier wird der Prozess erst gestartet, wenn der Browser bereit ist, den nächsten Frame der Animation zu zeichnen. Ein weiterer Vorteil ist, dass kein Zeichnen stattfindet, wenn der Ausschnitt gerade nicht sichtbar ist.

Es kann sinnvoll sein, länger laufende JavaScripts über das Web Worker API vom Hauptthread in einen Hintergrundthread zu verschieben.

Neben JavaScript gibt es auch die Möglichkeit, Animationen über CSS umzusetzen – das heißt, für einfache Fälle die Eigenschaft *transition* zu nutzen und für komplexere Anforderungen die Animationen über *@keyframes/animation* umzusetzen. Wenn möglich, sollten Sie der CSS-Variante den Vorzug vor JavaScript geben. Hilfreich können hier Bibliotheken wie css-animations.js sein.

Performante Eigenschaften

Animationen basieren auf der zeitgesteuerten Änderung von CSS-Eigenschaften. Häufig lassen sich dieselben Effekte mit verschiedenen CSS-Eigenschaften umsetzen, wobei es jedoch Unterschiede bei der Performance gibt.

Sie können ein Element bewegen, indem Sie auf *position: absolute* setzen und die Werte für *top* und *left* verändern. Im Beispiel soll ein Element mit der Klasse *.eins* beim Hovern nach unten links verschoben werden:

```
.eins {
  position: absolute;
  top: 0;
  left: 0;
  transition: all 1s ease;
}
.eins:hover {
  top: 2em;
  left: 2em;
}
```

Dieselbe Wirkung können Sie aber auch über *transform: translate()* erzielen, was bei einem Element mit der Klasse *.zwei* eingesetzt wird:

```
.zwei {
  transition: all 1s ease;
}
.zwei:hover {
  transform: translate(2em, 2em);
}
```

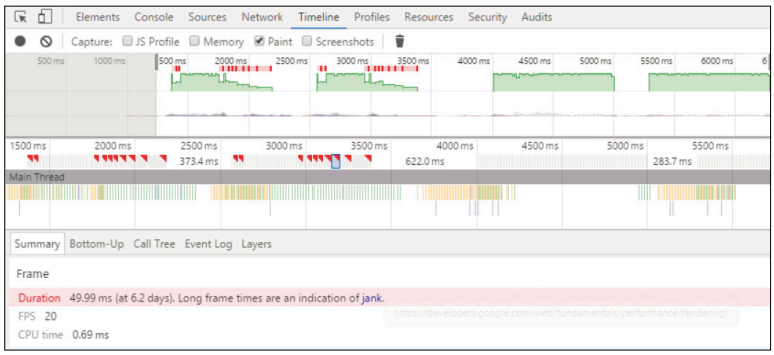
Die bessere Variante lässt sich über die Timeline der Chrome Entwickler-Tools ermitteln (Bild 4): Bei einer Aufzeichnung beider Interaktionen zeigen sich ganz deutlich mehrere rote Markierungen, wenn die Positionierungsänderung über *position: absolute* verändert wird. Das sind Warnungen vor den zu langen Frames, die für ein Ruckeln bei der Animation sorgen können. Die Transition mit *transform* zeigt hingegen keine solche Markierungen.

Auch bei Größenveränderungen von Elementen stehen Ihnen mehrere Möglichkeiten offen: Zum einen können Sie die *width*- und *height*-Angaben ändern oder alternativ auf *transform:scale()* zurückgreifen – auch in diesem Fall ist *transform* die eindeutig bessere Variante.

Den Grund für diese Unterschiede verrät die bereits erwähnte Webseite CSS Triggers: Bei *top*, *left*, *width* oder *height* sind alle drei möglichen Bereiche – Layout, Paint, Composite – betroffen, bei *transform* ist es hingegen nur Composite. Ebenfalls gut für Animationen geeignet ist die Eigenschaft *opacity*. Wo immer möglich, sollten Sie deswegen bei Animationen *opacity* oder *transform* einsetzen.

Aber auch Animationen von scheinbar ganz anderer Art lassen sich durch *opacity* und *transform* umschreiben, was Tobias Ahlin am Beispiel zeigt: Hier geht es um animierte Buttons, die auf den ersten Blick eine Animation über *box-shadow* nahelegen. Besser für die Performance ist es aber, hier mit *opacity* zu tricksen.

Im Beispiel kann man den beim Hovern gewünschten Endzustand einem weiteren, als erzeugter Inhalt eingeführten Element zuweisen. Dieses Element wird zuerst mit *opacity: 0* unsichtbar und ändert beim Hovern dann die Sichtbarkeit. Damit haben wir nur zwei distinktive *box-shadow*-Zustände, die nicht animiert werden – animiert wird nur die *opacity*. Damit die Animationen optimal ablaufen, muss sich das Element allerdings auf einer eigenen Ebene befinden. Dafür verwendet man klassisch *transform: translateZ(0)*. Besser ist für neuere Browser die CSS3-Eigenschaft *will-change*, die über kommende Änderungen informiert.

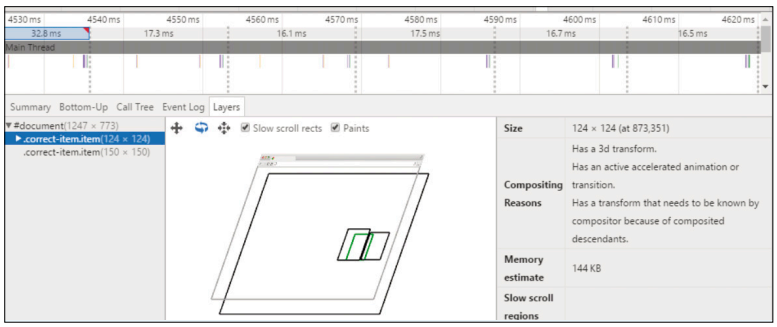


Rot markiert und mit Warnungen versehen ist die Animation, die über die Eigenschaft *position* realisiert wird (Bild 4)

Kommende Änderungen

Bei Operationen wie beispielsweise 3D-Transformationen werden die Elemente auf eine eigene Ebene bewegt (Bild 5), die unabhängig vom Rest der Seite dargestellt werden kann und später dann *composited*, das heißt ausgegeben werden kann. Das bringt einen eindeutigen Vorteil für die Performance.

Um ein Element explizit auf eine eigene Ebene zu verschieben, greift man gerne auf *translateZ()* oder *translate3D()* zurück. Auch wenn bei einem Element keine 3D-Transformationen stattfinden,



In der Timeline von Chrome sieht man auch, wenn sich einzelne Paint-Vorgänge auf eigenen Ebenen befinden (Bild 5)

gibt man etwa an: *transform: translate3d(0,0,0);*. Dies hat keinen sichtbaren Effekt, es bewirkt nur, dass das Element auf eine eigene Ebene verschoben wird. Andererseits ist aber auch das Verschieben auf eine eigene Ebene kostenintensiv.

Prinzipiell ist es besser, wenn ein Stylesheet-Autor keine Performance-Hacks anwendet, sondern die Optimierung der Performance dem Browser überlässt. Und genau dafür gibt es die neue CSS3-Eigenschaft *will-change*. Diese informiert den Browser, welche Art von Änderung stattfinden wird, sodass der Browser sie vorbereiten kann. Diese neue Eigenschaft ist in den aktuellen Browsern relativ gut implementiert – mit Ausnahme von IE/Edge (Bild 6).

Folgende Werte sind bei *will-change* möglich:

- *will-change: auto*: Damit überlässt man dem Browser alle Optimierungen.
- *will-change: scroll-position* bedeutet, dass sich die Scrollposition des Elements bald ändert.
- *will-change: contents* steht dafür, dass der Autor des Stylesheets davon ausgeht, dass sich etwas am Inhalt des Elements ändert.

Außerdem können die CSS-Eigenschaften aufgeführt werden, die verändert werden.

Sparsam und gezielt

Um ruckelfreie Animationen zu garantieren, könnte man versucht sein, *will-change global* für alle Elemente einzusetzen. Aber das wäre kontraproduktiv. Das globale Anwenden von *will-change* führt zu einer Verlangsamung der Seite oder sogar zu einem Absturz. Deswegen ist es wichtig, *will-change* gezielt nur dort zu benutzen, wo es auch benötigt wird.

Damit *will-change* wie gewünscht funktioniert, muss es außerdem zum Einsatz kommen, bevor die Aktion stattfindet, und nicht währenddessen. Wenn es beim Hovern Änderungen gibt, bringt es nichts, beim Hoverzustand *will-change* zu schreiben. Sinnvoller ist es dann, *will-change* beim Hovern über das Elternelement anzugeben:

```
.element {
  transition: opacity .2s;
  opacity: 1;
}
.container:hover > .element {
  will-change: opacity;
}
.element:hover {
  opacity: .3;
}
```

Außerdem sollte *will-change* entfernt werden, nachdem die Änderung stattgefunden hat. Am besten setzt man es deswegen mit JavaScript ein. Im folgenden Beispiel gibt es die Funktionen *hintBrowser()* und *removeHint()*, die die Eigenschaft *will-change* dynamisch bei Bedarf setzen – im Beispiel wird es beim Bewegen in einen Bereich gesetzt und am Ende der Animation wieder entfernt:

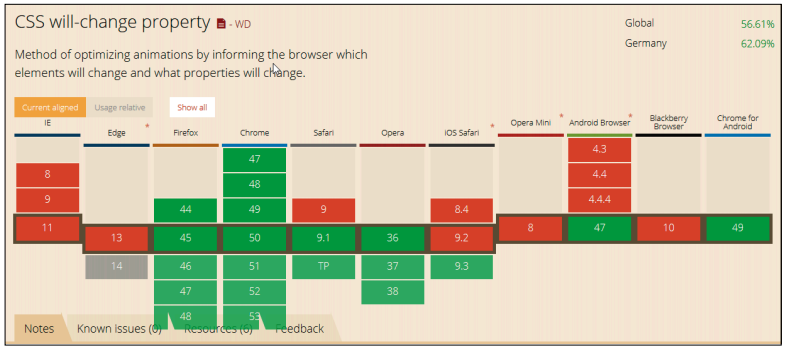
```
var el = document.getElementById('element');
el.addEventListener('mouseenter', hintBrowser);
el.addEventListener('animationEnd', removeHint);
function hintBrowser() {
  this.style.willChange = 'transform, opacity'; }
function removeHint() {
  this.style.willChange = 'auto'; }
```

Es gibt einen weiteren Grund, warum Sie *will-change* nicht global einsetzen sollten. Wenn nämlich die Eigenschaft, die bei *will-change* angegeben ist, bestimmte Nebenwirkungen hat, so werden diese Nebenwirkungen bereits durch *will-change* ausgelöst, auch wenn die eigentliche Änderung noch gar nicht stattgefunden hat.

Zu diesen Nebenwirkungen zählen die Erzeugung eines neuen Stapelkontexts oder eines Containing Blocks. Stapelkontexte sind für die Auswirkung der Eigenschaft *z-index* entscheidend, und Containing Blocks bestimmen den Bezugspunkt einer Positionierung etwa über *position: absolute*:

```
<div class="container">
  <p class="positioniert">!</p>
</div>
```

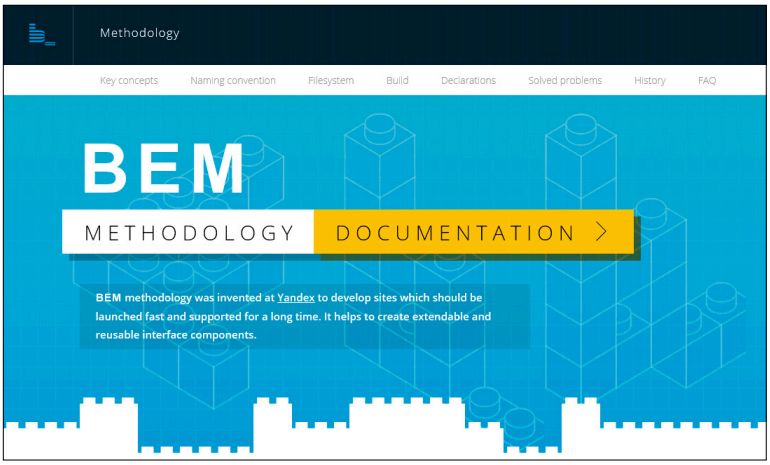
Das Element *.positioniert* wird absolut mit Hilfe von CSS positioniert: ▶



Browserunterstützung für die neue Eigenschaft *will-change:* Bis auf den Internet Explorer/Edge sieht es gut aus (Bild 6)

```
.positioniert {
  position: absolute;
  top: 0px;
  left: 0px;
}
```

In diesem Fall – wenn keine weitere Angaben gegeben werden – ist das Browserfenster der Bezugspunkt für die Positionierung. Wenn man das Element *.positioniert* hingegen in Bezug auf den *.container* platzieren möchte, könnte man den *.container* relativ positionieren. Auch *transform* hat diese Auswirkung. Es erstellt einen neuen Containingblock und bildet damit den Bezugspunkt für die Positionierung von *.positioniert*:



BEM: So präsentiert sich die BEM-Projektseite (Bild 7)

```
.container {
  transform: rotate(0deg);
}
```

Dieselbe Auswirkung hat jetzt auch die CSS-Anweisung *will-change: transform*:

```
.container {
  will-change: transform;
}
```

Damit verhält sich *.container* für die Positionierung der darin befindlichen Elemente so, als wäre *transform* angewandt. Das ist eine Nebenwirkung von *will-change* und bedeutet, dass es – global angewendet – jedes Layout zerstören würde.

Besser einfach

Es gibt noch weitere Faktoren im Stylesheet, die eine Rolle bei der Rendering-Performance spielen können. Bei Änderungen muss der Browser immer ermitteln, für welche Elemente eine Formatierung gilt, und dann das Ergebnis der Formatierung berechnen. Beides können intensive Operationen sein, die Sie dem Browser nicht unnötig schwermachen sollten. Verwenden Sie daher möglichst einfache Selektoren und beschränken Sie die Anzahl der formatierten Elemente, wo immer es geht. Ein Selektor wie der folgende ist für den Browser relativ kompliziert, weil er die Struktur berücksichtigen muss, um die zutreffenden Elemente zu ermitteln:

```
.box:nth-last-child(-n+1) .content {
  /* Formatierungen */
}
```

Einfacher sind hier Klassenselektoren:

```
.final {
  /* Formatierungen */
}
```

Das Problem an dieser Herangehensweise ist jedoch, dass ein einzelner Klassenselektor wie im Beispiel nichts über die Struktur des Dokuments verrät, was die Wartung erschwert. Abhilfe bietet an dieser Stelle BEM (Bild 7). BEM steht für Block, Element, Modifier und ist eine vom Yandex-Team entwickelte Methode, um CSS-Selektoren in HTML/CSS zu benennen.

Bei BEM werden für Blockelemente einfache Selektoren wie beispielsweise *.btn* eingesetzt. Elemente, die sich innerhalb des Blockelements befinden, erhalten dann einen Selektor, der aus dem Blockelement besteht, gefolgt von zwei Unterstrichen und dem eigentlichen Element: *.btn__price {}*. Außerdem gibt es noch die Modifizierer, die nach zwei Bindestrichen notiert werden, also beispielsweise *.btn—big {}*.

Im HTML-Code werden die Klassen verwendet und typischerweise auch mehrere Klassen:

```
<a class="btn btn-big" href="#">
  <span class="btn__price">23 €</span>
</a>
```

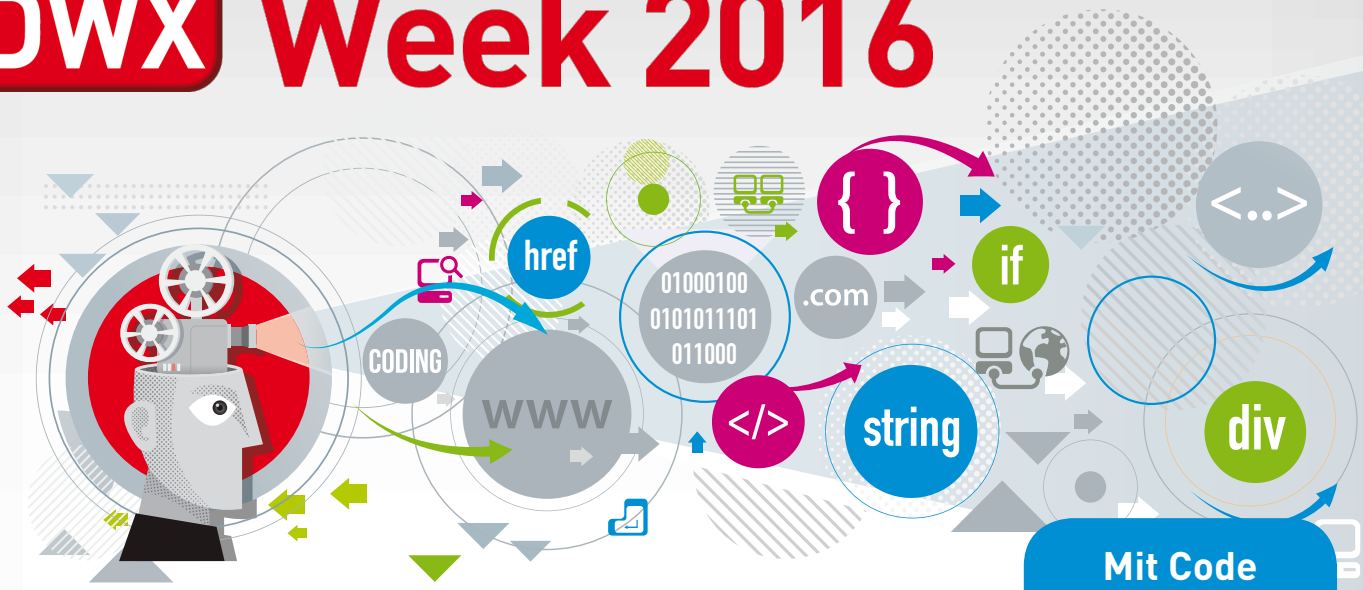
Der Vorteil dieser Technik: Wir setzen immer auf einfache Klassen, was dem Browser das Ermitteln der Elemente erleichtert, gleichzeitig sind diese Selektoren aber gut zu lesen und veranschaulichen die Struktur. Weitere Performance-Verbesserungen können dadurch erzielt werden, dass man etwa auf Flexbox statt auf das klassische *float* für Layouts und Layoutanpassungen setzt.

Prinzipiell sollten Sie aber immer prüfen, inwieweit einzelne Verbesserungen sich auswirken. Denn das Gute ist, dass Browser selbst ihre Performance ständig verbessern. ■

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

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

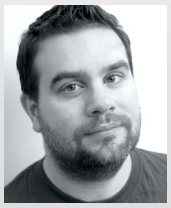
Peter Kröner

Stefan Lieser

Neno Loje

Bernd Marquardt

Pascal Precht


Golo Roden

Dr. Holger Schwichtenberg

Manfred Steyer

Ralf Westphal

developer-week.de DeveloperWeek

Aussteller & Sponsoren:

adesso · adorsys · AIT · brainLight · DevExpress · Fast Reports · GFU Cyrus AG · HANSER · HeiReS · hemmersbach · JetBrains · MATHEMA · MAXIMAGO · Microsoft · Saxonia Systems · SDX · SENACOR · SHS VIVEON · SOS · Telerik · TEXT CONTROL · ThingWorx · hello, technology · UID

Veranstalter:
developer media
Neue Mediengesellschaft Ulm mbH

Präsentiert von:
dotnetpro
web & mobile DEVELOPER

OBJEKTORIENTIERUNG IN JAVASCRIPT

Bauplan für Objekte

In JavaScript sieht Objektorientierung anders aus als in Sprachen wie Java.

Bevor wir uns der Objektorientierung in JavaScript zuwenden, zunächst eine kurze Wiederholung der wesentlichen (aber nicht aller) Prinzipien, die der Objektorientierung zugrunde liegen:

- Abstraktion: abstraktes Verhalten von Objekten wird in Klassen oder Prototypen zusammengefasst.
- Datenkapselung: Eigenschaften und Methoden werden in Form von Klassen oder Prototypen gekapselt und vor dem Zugriff von außen verborgen.
- Vererbung: Eigenschaften und Methoden können von einer Klasse an eine andere Klasse oder von einem Objekt (dem Prototypen) an ein anderes Objekt vererbt werden.
- Polymorphie: Objekte können abhängig von ihrer Verwendung unterschiedliche Typen annehmen.

Unter einer Klasse versteht man in der Objektorientierung bekanntermaßen eine Art Bauplan für Objekte. Eine Klasse dient als Vorlage, aus der zur Laufzeit eines Programms einzelne Instanzen (Objektinstanzen) erstellt werden können. Innerhalb einer Klasse definiert man die Eigenschaften und Methoden, über die die einzelnen Objektinstanzen verfügen sollen. Die Eigenschaften repräsentieren dabei den Zustand der Objektinstanzen, die Methoden deren Verhalten.

So weit, so gut. Doch JavaScript ist keine klassenbasierte Programmiersprache. Daran ändert auch nichts die in ES6/ES2015 eingeführte Klassensyntax, die oberflächlich Ähnlichkeiten zu der Syntax aufweist, wie sie beispielsweise von Java verwendet wird, und daher für Spracheinsteiger den Schluss aufkommen lassen könnte, JavaScript sei klassenbasiert.

Doch JavaScript ist eine sogenannte objektbasierte Programmiersprache. Objektinstanzen werden hier nicht auf Basis von Klassen, sondern auf Basis von anderen Objekten (den Prototypen) erstellt.

Wegen des Prinzips der Prototypen wird die Art der Programmierung in JavaScript auch als prototypenbasierte Programmierung oder als prototypische Programmierung bezeichnet. Eine alternative Bezeichnung ist auch klassenlose Programmierung, weil eben keine Klassen zum Einsatz kommen.

Objektorientierung in JavaScript

Dennoch verhält sich JavaScript bezüglich der objektorientierten Programmierung nicht so eindeutig wie andere Sprachen. Vielmehr gibt es in JavaScript verschiedene Techniken, um objektorientiert zu programmieren. Prinzipiell unterscheidet man dabei zwischen folgenden Techniken:

- Prototypische Objektorientierung: Dies ist die Art der Objektorientierung, die der Sprache JavaScript am natürlichsten ist, weil hierbei lediglich Objekte zum Einsatz kommen.
- Pseudoklassische Objektorientierung: Hierbei handelt es sich um eine Art der Objektorientierung, bei der man so tut, als sei JavaScript eine klassenbasierte Programmiersprache. Zum Einsatz kommen hierbei Konstruktorfunktionen.
- Objektorientierung mit Klassensyntax: Diese Art der Objektorientierung stellt eine syntaktische Vereinfachung der pseudoklassischen Objektorientierung dar und wurde mit ES6/ES2015 eingeführt.

Betrachten wir im Folgenden diese verschiedenen Arten etwas näher.

Prototypische Objektorientierung

Jedes Objekt in JavaScript basiert auf einem Prototyp. Zumindest fast jedes Objekt, denn einige Objekte, wie beispielsweise die Mutter aller Objekte, *Object,* oder Objekte, deren Prototyp explizit auf null gesetzt wurde, haben keinen Prototyp.

Jedes Objekt kann zudem selbst als Vorlage, also als Prototyp für ein anderes Objekt herhalten. In diesem Fall erbt das neue Objekt die Eigenschaften und Methoden vom Prototyp.

Listing 1: Prototypische Vererbung

```
'use strict';
let animal = {
  name: '',
  color: 'Braun',
  age: 0,
  eat: function(food) {
    console.log('Mmpf mmpf', + food + '!');
  },
  drink: function(drink) {
    console.log('Mmmmmmh', + drink + '!');
  }
}
let cat = Object.create(animal);
cat.eat('Katzenfutter');
cat.drink('Milch');
console.log(cat.color);
let dog = Object.create(animal);
dog.eat('Fleisch');
dog.drink('Wasser');
console.log(dog.color);
```

Listing 2: Erbende Objekte

```
'use strict';
let animal = {
  name: 'default',
  color: 'Braun',
  age: 0,
  eat: function(food) {
    console.log('Mmpf mmpf', + food + '!');
  },
  drink: function(drink) {
    console.log('Mmmmmmh', + drink + '!');
  }
}
let cat = Object.create(animal);
cat.meow = function() {
  console.log('Miauuuuuu!');
}
cat.eat('Katzenfutter');
cat.drink('Milch');
cat.meow();
console.log(cat.color);
let dog = Object.create(animal);
dog.bark = function() {
  console.log('Wuff wuff!');
}
dog.eat('Fleisch');
dog.drink('Wasser');
dog.bark();
console.log(dog.color);
```

Übersetzt in die klassenbasierte Programmierung entspricht das also in etwa dem Erstellen einer Unterklasse. Diese erbt ja auch von der abgeleiteten (Ober-)Klasse (beziehungsweise Superklasse oder Elternklasse) einen Teil ihrer Eigenschaften und Methoden.

Um ein Objekt auf Basis eines Prototyps zu definieren, verwendet man in JavaScript die Methode *Object.create()*. Ein Beispiel hierfür zeigt Listing 1: Hier wird zunächst das Objekt *animal* definiert, das über die Methode *eat()* und die Eigenschaften *name, color* und *age* verfügt (auf zusätzliche Setter und Getter für die Eigenschaften wurde an dieser Stelle aus Platzgründen verzichtet). Anschließend werden über *Object.create()* zwei weitere Objekte erstellt (*cat* und *dog*), wobei hier jeweils das Objekt *animal* als Prototyp übergeben wird. Die beiden Objekte *cat* und *dog* erben die Methoden *eat()* und *drink()* sowie die Eigenschaften *name, color* und *age* von ihrem Prototyp.

Es ist selbstverständlich auch möglich, an Objekten, die von einem Prototyp erben, zusätzliche Eigenschaften und Methoden zu definieren. In Listing 2 beispielsweise erhält das Objekt *cat* die neue Methode *meow()* und das Objekt *dog* die neue Methode *bark()*.

Klassen beziehungsweise im Fall der prototypischen Objektorientierung Objekte, die in der Vererbungshierarchie weiter unten stehen, sind spezieller als solche, die in der Vererbungshierarchie weiter oben stehen. Oder umgekehrt formuliert: Klassen/Objekte, die in der Hierarchie weiter oben stehen, sind abstrakter als solche weiter unten.

Spezielleres Verhalten kann dabei zum einen dadurch entstehen, indem Klassen beziehungsweise Objekten neue Methoden hinzugefügt werden, zum anderen aber auch, indem geerbte Methoden überschrieben werden.

Ein Beispiel hierfür sehen Sie in Listing 3. Der Quelltext wurde hier um ein neues Objekt *vegetarianDog* erweitert, welches von *dog* ableitet. Die Methode *eat()* wurde hier entsprechend angepasst und überschrieben: sie liefert für den Fall, dass die der Methode übergebene Zeichenkette die Werte *Fleisch* oder *fleisch* enthält, einen entsprechenden Fehler.

Ruft man also im folgenden die Methode *eat()* auf dem Objekt *vegetarianDog* auf, wird genau diese Methode verwendet und nicht die Methode, die im *animal*-Objekt definiert ist.

Vererbung über die Prototypenkette

Objekte erben Eigenschaften und Methoden von ihrem Prototyp beziehungsweise allen vorangegangenen Prototypen über die sogenannte Prototypenkette. Vom Ablauf her ▶

Listing 3: Überschreiben einer Methode

```
'use strict';
let animal = {
  name: '',
  color: 'Braun',
  age: 0,
  eat: function(food) {
    console.log('Mmpf mmpf', + food + '!');
  },
  drink: function(drink) {
    console.log('Mmmmmmh', + drink + '!');
  }
}
let cat = Object.create(animal);
cat.meow = function() {
  console.log('Miauuuuuu!');
}
let dog = Object.create(animal);
dog.bark = function() {
  console.log('Wuff wuff!');
}
let vegetarianDog = Object.create(dog);
vegetarianDog.eat = function(food) {
  if(food.indexOf('Fleisch') >= 0 ||
  food.indexOf('fleisch') >= 0) {
    throw new Error('Ich esse doch kein Fleisch!');
  } else {
    console.log('Mmpf mmpf', + food + '!');
  }
}
vegetarianDog.eat('Käse');
```

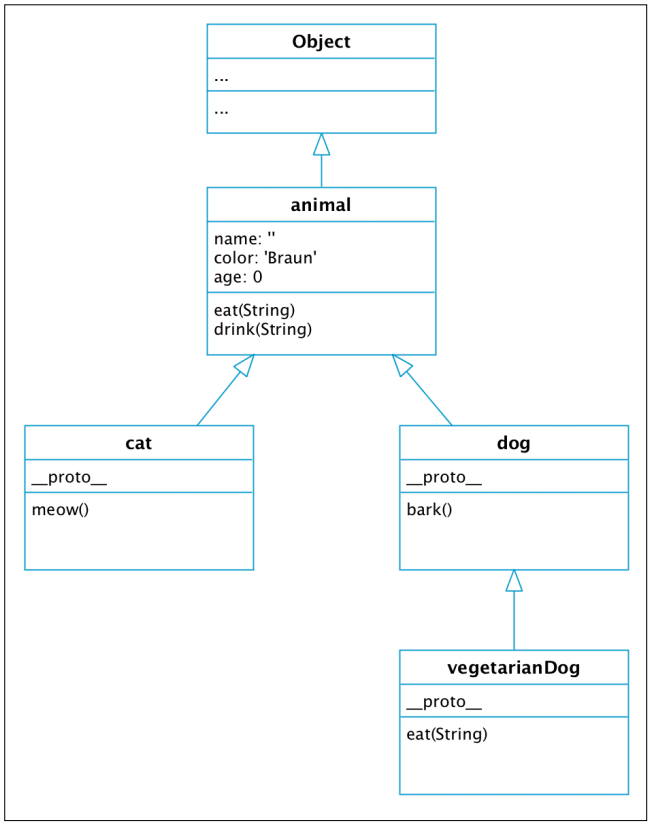
funktioniert das wie folgt: Wird auf eine Eigenschaft beziehungsweise Methode eines Objekts zugegriffen, prüft der JavaScript-Interpreter zunächst, ob das entsprechende Objekt über die jeweilige Eigenschaft/Methode verfügt. Ist dies der Fall, wird die Eigenschaft/Methode verwendet, andernfalls wird über auf den Prototyp des Objekts zugegriffen. Ist die Eigenschaft/Methode auch dort nicht vorhanden, wird im Prototyp des Prototyps geschaut, bis irgendwann das Basisobjekt *Object* erreicht wird (Bild 1).

Bezogen auf das Beispiel von eben, das in Form eines leicht abgewandelten Objektdiagramms dargestellt ist, bedeutet dies beispielsweise: Wenn man auf dem Objekt *vegetarianDog* die Methode *eat()* aufruft, wird nicht in der Prototypenkette nach oben gegangen, weil diese Methode direkt an dem Objekt definiert ist.

Ruft man dagegen die Methode *bark()* auf, geht es über die Prototypenkette eine Ebene hoch zum Objekt *dog*. Ruft man die Methode *drink()* auf, geht es zwei Ebenen hoch bis zu dem Objekt *animal*. Ruft man dagegen die Methode *toString()* auf, geht es sogar hoch bis zum Objekt *Object*, weil erst dort diese Methode definiert ist:

```
vegetarianDog.eat('Käse'); // Vorhanden in vegetarianDog
vegetarianDog.bark(); // Vorhanden in dog
vegetarianDog.drink('Wasser');  // Vorhanden in animal
vegetarianDog.toString(); // Vorhanden in Object
```

Um eine Methode des Prototyps eines Objekts aufzurufen, muss man zunächst an den Prototyp des Objekts gelangen.



Beispiel für eine Prototypenkette (Bild 1)

Listing 4: Konstruktorfunktionen

```
function Animal(name, color, age) {
  this.name = name;
  this.color = color;
  this.age = age;
};
Animal.prototype.eat = function(food) {
  console.log('Mmpf mmpf' , + food);
}
Animal.prototype.drink = function(drink) {
  console.log('Mmmmmmh' , + drink);
}
function Dog(name, color, age, type) {
  Animal.call(this, name, color, age);
  this.type = type;
};
Dog.prototype = new Animal();
Dog.prototype.constructor = Dog;
Dog.prototype.bark = function() {
  console.log('Wuff wuff');
}
let bello = new Dog('Bello', 'Weiß', 2, 'Malteser');
bello.bark();
```

Dazu hat man prinzipiell zwei Möglichkeiten: Zum einen ist der Prototyp eines Objekts in dessen Eigenschaft *__proto__* hinterlegt, zum anderen kann über die Methode *Object.getPrototypeOf()* der Prototyp eines Objekts ermittelt werden.

Stellt sich die Frage, wie die vier eingangs vorgestellten Grundprinzipien der objektorientierten Programmierung in der prototypischen Objektorientierung umgesetzt sind.

Das erste Prinzip, die Abstraktion von Verhalten, das verschiedenen Objektinstanzen gemeinsam ist, ist dadurch gegeben, dass dieses Verhalten in Prototypen implementiert wird. Das Verhalten wird zudem über die Prototypenkette an ableitende Objekte vererbt (insofern ist auch das Prinzip der Vererbung gegeben).

Eine echte Datenkapselung ist in JavaScript nur über Entwurfsmuster möglich. Es ist jedoch möglich, über die Schlüsselwörter *get* und *set* Zugriffseigenschaften zu definieren und damit auch den Zugriff auf Dateneigenschaften in gewisser Weise zu steuern oder aber durch Eigenschaftsattribute weitere Konfigurationen von Eigenschaften vorzunehmen.

Polymorphismus ist in JavaScript ohnehin schon gegeben, da es keine eingebauten Restriktionen hinsichtlich der Typen gibt, die zum Beispiel als Argumente für Funktionen verwendet werden.

Pseudoklassische Objektorientierung

Im Gegensatz zur prototypischen Objektorientierung basiert die pseudoklassische Objektorientierung auf dem Einsatz von Konstruktorfunktionen. Die Idee dabei ist es, dass sich die objektorientierte Programmierung mit JavaScript dann so anfühlt, als würde man klassenbasiert programmieren.

Zur Erinnerung: Bei Konstruktorfunktionen handelt es sich zunächst einmal um nichts anderes als normale Funktionen. Sie werden erst dadurch zu einer Konstruktorfunktion, dass man sie mit dem vorangestelltem *new*-Schlüsselwort aufruft. Zudem werden sie in der Regel (und den Konventionen der guten Software-Entwicklung folgend) in Upper-Camel-Case-Schreibweise geschrieben, um sie von normalen Funktionen zu unterscheiden. Und schließlich bezieht sich innerhalb einer Konstruktorfunktion das Schlüsselwort *this* auf das Objekt, das durch den Aufruf der Konstruktorfunktion erzeugt wird:

```
function Animal(name, color, age) {
  this.name = name;
  this.color = color;
  this.age = age;
};
```

Um unter Verwendung einer Konstruktorfunktion eine neue Objektinstanz zu erzeugen, ruft man die Konstruktorfunktion wie bereits gesagt mit dem Schlüsselwort *new* auf. Im folgenden Listing wird beispielsweise eine neue Instanz von *Animal* erzeugt, wobei als Argumente die Werte *Fischi*, *Grün* und *2* übergeben und in der erzeugten Instanz als Werte für die entsprechenden Eigenschaften gesetzt werden:

```
let fish = new Animal('Fischi', 'Grün', 2);
console.log(fish.name);

console.log(fish.color);
console.log(fish.age);
```

Um Methoden zu erstellen, die allen Objektinstanzen, die mit Hilfe einer Konstruktorfunktion erstellt werden, gemeinsam sind, definiert man diese an dem Objekt, das den durch die Konstruktorfunktion erzeugten Objektinstanzen als Prototyp dient. Dieses Objekt ist in der Eigenschaft *prototype* der entsprechenden Konstruktorfunktion hinterlegt.

Im folgenden Listing werden beispielsweise auf diese Art die zwei Methoden *eat()* und *drink()* an dem Prototypobjekt der Konstruktorfunktion *Animal* definiert. Anschließend lassen sich beide Methoden an der Objektinstanz *fish* aufrufen:

```
Animal.prototype.eat = function(food) {
  console.log('Mmpf mmpf' , + food + '!');
}
Animal.prototype.drink = function(drink) {
  console.log('Mmmmmmh' , + drink + '!');
}
let fish = new Animal('Fischi', 'Grün', 2);
fish.eat('Algen');
fish.drink('Wasser');
```

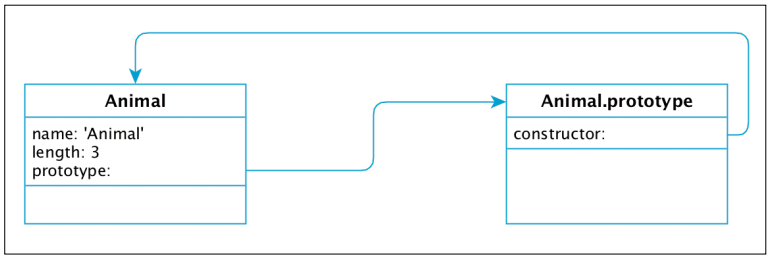
Das Prinzip der Vererbung bei der pseudoklassischen Vererbung ist nicht ganz so einfach zu verstehen, deshalb wird das folgenden Code-Beispiel durch Klassendiagramme, die den Zustand der JavaScript-Objekte nach entsprechenden Codezeilen zeigen, illustriert.

Zunächst wird in Listing 4 die Funktion *Animal* definiert, die als Konstruktorfunktion dienen soll (Schritt 1). Implizit passiert hierdurch aber wesentlich mehr: Der Code *function Animal() {...}* bewirkt zusätzlich, dass ein Objekt erstellt wird, das in der Eigenschaft *prototype* der Konstruktorfunktion hinterlegt ist und seinerseits über eine Eigenschaft *constructor* auf die Konstruktorfunktion verlinkt (Bild 2).

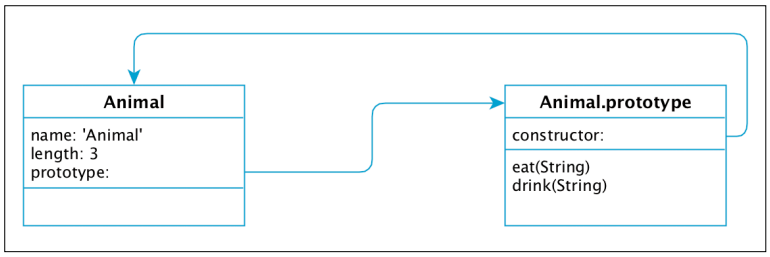
Das Objekt *Animal.prototype* ist der Prototyp für die Objekte, die durch Aufruf der Konstruktorfunktion *Animal* erstellt werden. Im zweiten Schritt werden nun die Methoden an diesem Prototyp definiert, im Beispiel die Methode *eat()* und *drink()* (Bild 3).

Im dritten Schritt wird es nun bezüglich der Vererbung interessant: Jetzt geht es nämlich daran, eine Unterklasse von *Animal* zu erzeugen. Dazu wird im Beispiel die Konstruktorfunktion *Dog* definiert. Bild 4 zeigt den Zustand danach.

Wie man sieht, besteht erst einmal noch keine Verknüpfung zwischen der *Dog*-Klasse und der *Animal*-Klasse. Diese Verknüpfung muss man als Entwickler manuell herstellen. Und zwar geschieht dies durch eine Verbindung zwischen den Prototyp-Objekten, sprich einer Verbindung von *Dog.prototype* zu *Animal.prototype*. Die Zeile *Dog.prototype = new Animal();* in Schritt 4 sorgt dabei zunächst dafür, dass *Dog.prototype* ein komplett neues Objekt referenziert, das seinerseits als Prototyp das Objekt *Animal.prototype* referenziert. Den jetzigen Zustand der Objekte zeigt Bild 5. Hier sieht man auch, dass das neue Objekt *Dog.prototype* im Vergleich zum alten Objekt *Dog.prototype* über keine Eigenschaft *constructor* verfügt und somit keine Verbindung zum Objekt *Dog* besteht. Dies wird in Schritt 5 korrigiert (Bild 6). ▶



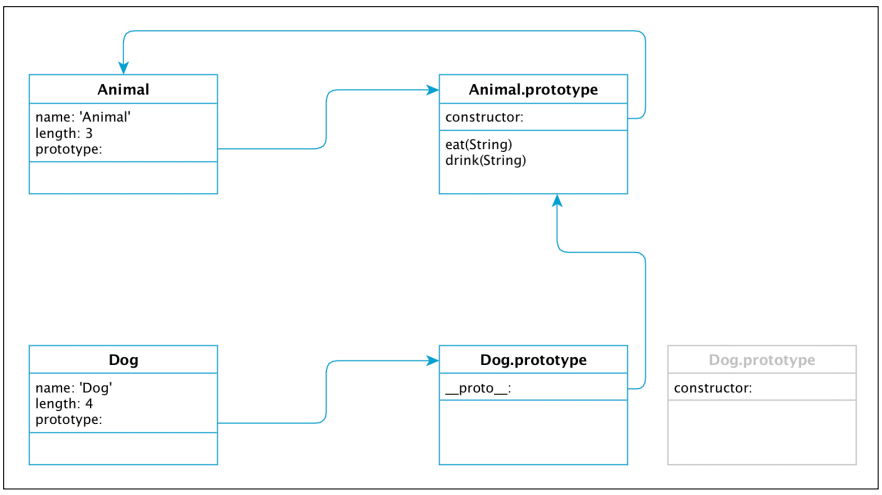
Schritt 1: Zustand nach Definition der Konstruktorfunktion *Animal* (Bild 2)



Schritt 2: Definition der Methode *eat()* am Prototyp (Bild 3)

Abschließend werden die Methoden der Unterklasse, sprich Methoden am Objekt *Dog.prototype*, definiert (Schritt 6). Hierbei ist es wichtig, das Hinzufügen von Eigenschaften und Methoden erst dann durchzuführen, nachdem *Dog.prototype* neu definiert wurde (siehe Schritt 4), da diese ansonsten am alten Objekts *Dog.prototype* definiert würden und damit verloren gingen.

Bild 7 zeigt also auch den finalen Zustand der Objekte: *Dog* und *Animal* als Konstruktorfunktionen (beziehungsweise Klassen), über die sich neue Objektinstanzen auf Basis der Prototypen *Dog.prototype* und *Animal.prototype* erzeugen lassen.



Schritt 4: Neudefinition des Dog-Prototyps (Bild 5)

Um den Konstruktor einer Oberklasse aufzurufen, muss man lediglich die entsprechende Konstruktorfunktion über die Methode *call()* aufrufen und dabei *this* als Ausführungskontext sowie optional weitere Parameter übergeben.

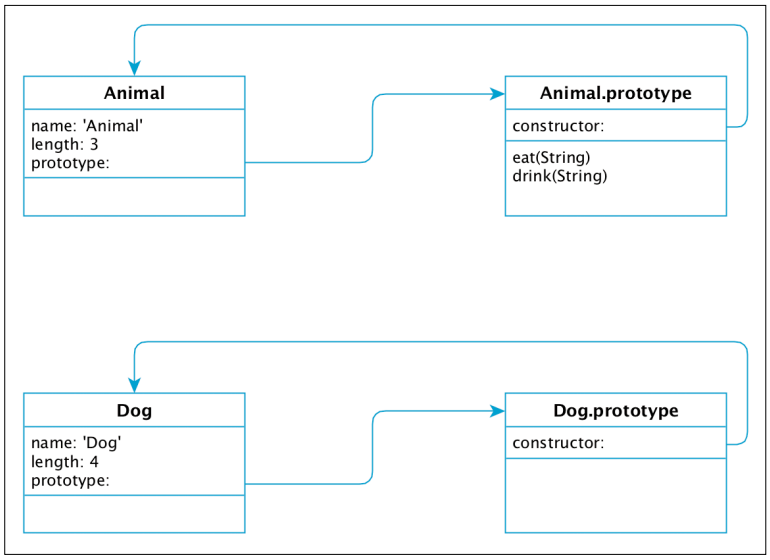
Das folgende Listing zeigt den entsprechenden Ausschnitt aus dem vorigen Listing, wobei aus der Konstruktorfunktion *Dog* die Konstruktorfunktion *Animal* aufgerufen wird:

```
function Dog(name, color, age, type)
{
  Animal.call(this, name, color, age);
  this.type = type;
};
```

Um eine Methode in einer Unterklasse zu überschreiben, definiert man die entsprechende Methode einfach an den Prototyp, der der entsprechenden Konstruktorfunktion der Unterklasse zugrunde liegt. Im folgenden Listing wird beispielsweise in der Klasse *VegetarianDog* die Methode *eat()* mit der aus den vorangegangenen Abschnitten bekannten Logik überschrieben:

```
function VegetarianDog(name, color, age, type) {
  Dog.call(this, name, color, age);
  this.type = type; };
VegetarianDog.prototype.eat = function(food) {
  if(food.indexOf('Fleisch') >= 0 || food.
indexOf('fleisch') >= 0) {
    throw new Error('Ich esse doch kein Fleisch!');
  } else {
    console.log('Mmpf mmpf' , + food); }
}
```

Der Aufruf einer Methode der Oberklasse funktioniert wie schon der Aufruf von Konstruktoren unter Verwendung der Methode *call()*. Im folgenden Listing wird beispielsweise die Methode *eat()* der Klasse *Animal* in der Methode *eat()* Klasse *VegetarianDog* aufgerufen:

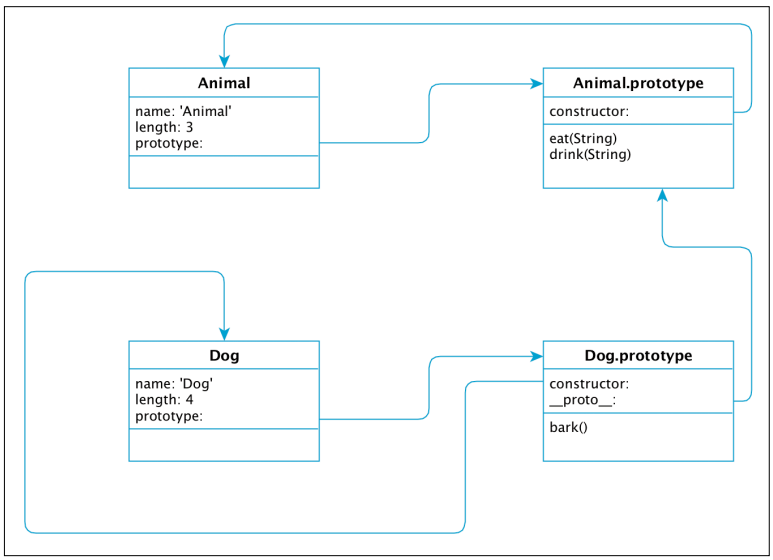


Schritt 3: die Konstruktorfunktion *Dog* (Bild 4)

```
function VegetarianDog(name,color,age,type) {
  Dog.call(this, name, color, age);
  this.type = type;
};
VegetarianDog.prototype.eat = function(food) {
  if(food.indexOf('Fleisch') >= 0 ||
  food.indexOf('fleisch') >= 0) {
    throw new Error('Ich esse doch kein
    Fleisch!');
  } else {
    Dog.prototype.eat.call(this, food);
  }
}
let vegetarianDog = new VegetarianDog('Bella',
'Weiß', 2, 'Malteser');
vegetarianDog.eat('Brot');
```

Zusammenfassend lässt sich für die pseudoklassische Objektorientierung Folgendes festhalten:

Abstraktion von Verhalten ist durch die Konstruktorfunktionen und den diesen zugrunde liegenden Prototyp gegeben. Vererbung kann auch bei der pseudoklassischen Objektorientierung realisiert werden, ist allerdings vergleichsweise aufwendig. Bezüglich der Datenkapselung und der Polymorphie gilt das Gleiche wie bei der prototypischen Objektorientierung. Allerdings ist der Aufwand für das Erstellen einer Klassenhierarchie vergleichsweise groß.



Schritt 6: Definition der Methode *bark()* am Prototyp (Bild 7)

Objektorientierung mit Klassensyntax

Seit ES6/ES2015 hat auch eine neue Klassensyntax in den ECMAScript-Standard Einzug gehalten. Diese Klassensyntax ermöglicht zwar wie bereits gesagt keine echte klassenbasierte Programmierung, fühlt sich aber aufgrund unter anderem der verwendeten Schlüsselwörter und Syntax relativ ähnlich an.

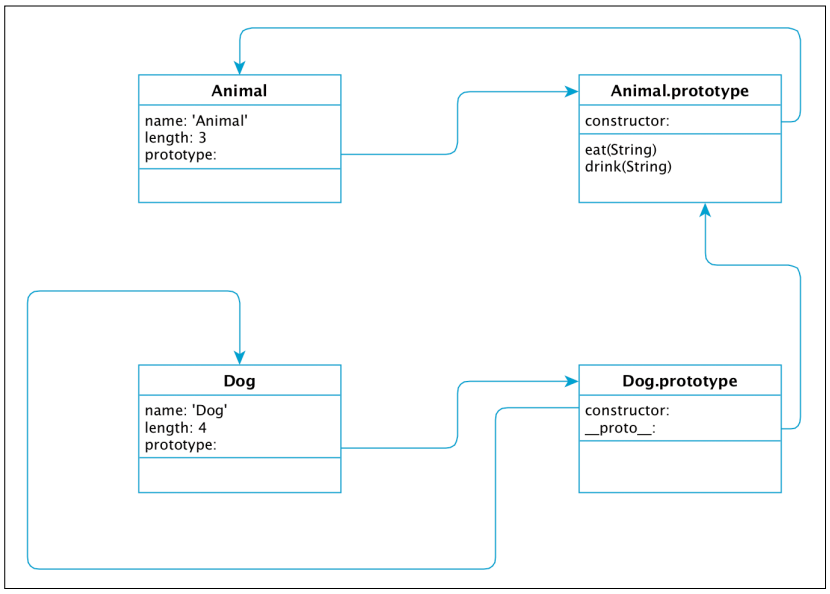
Um in JavaScript eine Klasse zu definieren, verwendet man das Schlüsselwort *class*. Hinter dieses Schlüsselwort schreibt man den Namen der Klasse, gefolgt von einem Paar geschweifter Klammern, die den Körper der Klasse definieren. Innerhalb dieses Klassenkörpers können anschließend die Methoden der Klasse definiert werden. Eigenschaften können dagegen an dieser Stelle nicht angegeben werden:

```
class Animal {
  constructor(name, color, age) {
    this.name = name;
    this.color = color;
    this.age = age;
  }
  eat(food) {
    console.log('Mmpf mmpf' , + food);
  }
  drink(drink) {
    console.log('Mmmmmmh' , + drink);
  }
  toString() {
    return this.name + '' , + this.color + '' ,
    + this.age;
  }
}
```

Eine besondere Rolle kommt dabei der Methode *constructor()* zu, denn diese Methode wird immer dann aufgerufen, wenn eine neue Objektinstanz der Klasse erzeugt wird. Vom Prinzip her entspricht das in etwa einer Konstruktorfunktion: Zurückgeliefert wird von *constructor()* nämlich implizit eine neue Objektinstanz, wobei innerhalb der Methode *constructor()* über das Schlüsselwort *this* auf diese Objektinstanz zugegriffen werden kann. Das ist wiederum dann auch die Stelle, an der Eigenschaften definiert werden können.

Auf die aktuelle Objektinstanz beziehungsweise die so definierten Eigenschaften lässt sich auch innerhalb anderer Methoden der Klasse über das Schlüsselwort *this* zugreifen, wie man am Beispiel der Methode *toString()* sehen kann. Um eine Objektinstanz einer Klasse zu erzeugen, verwendet man – wie von Konstruktorfunktionen bekannt – das Schlüsselwort *new:* Der Aufruf von *new Animal()* sorgt hier dafür, dass eine neue Instanz von *Animal* erzeugt wird, deren Eigenschaft *name* den Wert *Fischi* hat, die Eigenschaft *color* den Wert *Grün* und die Eigenschaft *age* den Wert 2:

```
let fish = new Animal('Fischi',
'Grün', 2);
```



Schritt 5: Verlinken vom Prototyp zur Konstruktorfunktion (Bild 6)

```
fish.eat('Algen');
console.log(fish.toString());
```

Die Schlüsselwörter *get* und *set*, über die (beispielsweise in der Objekt-Literal-Schreibweise) Getter- und Setter-Methoden (kurz: Getter und Setter) an Objekten definiert werden können, können auch in Kombination mit Klassen verwendet werden, um dort Getter und Setter zu definieren. Die beiden Schlüsselwörter schreibt man wie gewohnt vor den Namen der jeweiligen Methode.

Zu beachten ist außerdem, dass der Name der jeweiligen Getter- beziehungsweise Setter-Methode nicht gleich der entsprechenden Eigenschaft ist, um eine Endlosrekursion, einen damit verbundenen Stacküberlauf und damit den Absturz des jeweiligen Programms zu verhindern. Eine Möglichkeit dabei ist es, wie in Listing 5 dem Namen der Eigenschaft einen Unterstrich voranzustellen *(_name, _color, _age)*.

Listing 5: Getter und Setter

```
class Animal {
  constructor(name, color, age) {
    this.name = name;
    this.color = color;
    this.age = age;
  }
  eat(food) {
    console.log('Mmpf mmpf' , + food);
  }
  drink(drink) {
    console.log('Mmmmmmh' , + drink);
  }
  get name() {
    return this._name;
  }
  set name(name) {
    this._name = name;
  }
  get color() {
    return this._color;
  }
  set color(color) {
    this._color = color;
  }
  get age() {
    return this._age;
  }
  set age(age) {
    this._age = age;
  }
}
let snake = new Animal('Zischi', 'Grün', 5);
console.log(snake.name);
console.log(snake.color);
console.log(snake.age);
```

Über das Schlüsselwort *extends* ist es möglich, dass eine Klasse von einer anderen Klasse erbt beziehungsweise ableitet. Das Schlüsselwort schreibt man dabei einfach hinter die (neue) Klasse, die erben soll, dem Schlüsselwort folgt dann der Name der Klasse, von der geerbt werden soll.

Erzeugt man eine Instanz der Unterklasse, verfügt diese über alle in der Oberklasse definierten Methoden sowie die Methoden, die in der Unterklasse definiert wurden:

```
class Dog extends Animal {
  constructor(name, color, age, type) {
    super(name, color, age);
    this.type = type; }
  get type() {
    return this._type; }
  set type(type) {
    this._type = type; }
  bark() {
    console.log('Wuff wuff'); }
}

let dog = new Dog('Bello', 'Weiß', 2, 'Malteser');
dog.eat('Käse');
dog.bark();
```

Zugriff auf den den Konstruktor der Oberklasse (den Elternkonstruktor) geschieht über den Aufruf *super()*. Dabei ist es wichtig, diesen Aufruf zu tätigen, bevor man innerhalb des Konstruktors auf *this* zugreift. Ansonsten kommt es zu einem Fehler. Das Gleiche gilt, wenn man innerhalb eines Konstruktors den Aufruf des Elternkonstruktors ganz weglässt.

Konstruktor einer Klasse

Es ist allerdings möglich,den Konstruktor einer Klasse, sprich die *constructor()*-Methode, ganz wegzulassen. In diesem Fall wird intern implizit ein Konstruktor verwendet, der alle Parameter an den Elternkonstruktor weiterleitet:

```
class Dog extends Animal {
  // Implizit wird folgender Konstruktor verwendet:
  // constructor(... args) {
  //   super(...args);
  // }
  bark() {
    console.log('Wuff wuff');
  }
}
```

Die Struktur, die intern durch die Klassen *Animal* und *Dog* erzeugt wird, entspricht übrigens exakt der Struktur, die durch die vergleichbare pseudoklassische Objektorientierung erzeugt wurde.

Um eine Methode in einer Unterklasse zu überschreiben, definiert man einfach eine entsprechende Methode mit gleichem Namen wie die zu überschreibende Methode. Dazu ein Code-Beispiel: Ziel ist es, wie in den Beispielen für die anderen Arten der Objektorientierung eine neue Klasse *Vegetari-*

anDog zu erstellen, die von *Dog* ableitet und die Methode *eat()* überschreibt.

Um das Beispiel noch ein bisschen objektorientierter zu machen und die Methode *eat()* dahingehend zu ändern, dass sie nicht mehr eine Zeichenkette als Parameter erwartet, sondern eine Objektinstanz, wird zunächst eine einfache Klassenhierarchie für Essen erstellt (Listing 6). Diese besteht aus der Basisklasse *Food* und den beiden Unterklassen *Meat* und *Bread*. Als Parameter erwartet der Konstruktor eine Beschreibung des jeweiligen Essens in Form einer Zeichenkette. Die Methode *toString()*, die lediglich diese Beschreibung zurückgibt, gewährleistet außerdem, dass die Konsolenausgabe der Methode *eat()* der Klasse *Animal* nicht angepasst werden muss und weiterhin eine vernünftige Ausgabe erzeugt.

In der Klasse *VegetarianDog* wird innerhalb der Methode *eat()* nicht (wie in den Beispielen bei den anderen Arten der Objektorientierung) geprüft, ob die Werte *Fleisch* oder *fleisch* in einer Zeichenkette enthalten sind, sondern lediglich, ob es sich bei dem übergebenen Argument um eine Instanz der Klasse *Meat* handelt. Ist dies der Fall, wird ein Fehler geworfen, andernfalls eine entsprechende Ausgabe erzeugt:

```
class VegetarianDog extends Dog {
  constructor(name, color, age, type) {
    super(name, color, age, type);
  }
  eat(food) {
    if(food instanceof Meat) {
      throw new Error('Ich esse doch kein Fleisch!');
    } else {
      console.log('Mmpf mmpf', + food.description
      + '!');
    }
  }
}
let dog = new VegetarianDog('Bello', 'Weiß', 2,
'Malteser');
dog.eat(new Bread('Weizenbrot'));
dog.eat(new Meat('Steak'));
```

Listing 6: Ein einfaches Objektmodell

```
class Food {
  constructor(description) {
    this.description = description;
  }
  get description() {
    return this._description;
  }
  set description(description) {
    this._description = description;
  }
  toString() {
    return this.description;
  }
}
class Meat extends Food {
  constructor(description) {
    super(description);
  }
}
class Bread extends Food {
  constructor(description) {
    super(description);
  }
}
let bread = new Bread('Weizenbrot');
console.log(bread.description);
let meat = new Meat('Steak');
console.log(meat.description);
```

Anschließend lässt sich die Methode aufrufen, wobei jeweils Instanzen von Food (beziehungsweise deren Unterklassen) übergeben werden:

```
let dog = new VegetarianDog('Bello', 'Weiß', 2,
'Malteser');
dog.eat(new Bread('Weizenbrot'));

dog.eat(new Meat('Steak'));
```

Wenn man innerhalb einer Methode eine andere Methode der gleichen Klasse aufrufen möchte, so kann man dies über die Verwendung des Schlüsselworts *this* erreichen. Möchte man dagegen innerhalb einer Methode gezielt eine Methode der Oberklasse aufrufen, verwendet man das Schlüsselwort *super*, das sozusagen eine Referenz auf die Oberklasse darstellt.

Bezogen auf das Beispiel von eben könnte man beispielsweise in der Klasse *VegetarianDog* die Methode *eat()* der Oberklasse *Animal* wiederverwenden, nämlich genau für den Fall, dass es sich bei dem der Methode übergebenen Essen um kein Fleisch handelt.

Statische Methoden und statische Eigenschaften

Statische Methoden bezeichnen in der objektorientierten Programmierung solche Methoden, die man direkt auf der Klasse aufruft und nicht auf einer Objektinstanz. Statische Methode eignen sich immer dann, wenn man damit Funktionalität bereitstellen möchte, die thematisch zu der jeweiligen Klasse gehört, aber nicht zwangsweise eine Objektinstanz voraussetzt.

In der Klassensyntax in JavaScript definiert man eine Methode als statisch, indem man ihr das Schlüsselwort *static* voranstellt. Anschließend kann die Methode direkt über die Klasse aufgerufen werden. Listing 7 zeigt mit *getAnimalColors()* ein Beispiel für eine statische Methode: Diese Methode kann nur auf der Klasse *Animal* aufgerufen werden, nicht aber auf deren Objektinstanzen. Neben statischen Methoden gibt es in der Objektorientierung auch das Konzept der ▶

Listing 7: Statische Methode

```
class Animal {
  constructor(name, color, age) {
    this.name = name;
    this.color = color;
    this.age = age;
  }
  static getAnimalColors() {
    return {
      WHITE: 'Weiß',
      BLACK: 'Schwarz',
      BROWN: 'Braun',
      GREEN: 'Grün',
      YELLOW: 'Gelb',
      ORANGE: 'Orange'
    }
  }
  eat(food) {
    console.log('Mmpf mmpf' , + food);
  }
  drink(drink) {
    console.log('Mmmmmmh' , + drink);
  }
  get name() {
    return this._name;
  }
  set name(name) {
    this._name = name;
  }
  get color() {
    return this._color;
  }
  set color(color) {
    this._color = color;
  }
  get age() {
    return this._age;
  }
  set age(age) {
    this._age = age;
  }
}

console.log(Animal.getAnimalColors());
let bird = new Animal('Birdie',
Animal.getAnimalColors().BLACK, 5);
console.log(bird.name);
console.log(bird.color);
console.log(bird.age);

// console.log(bird.getAnimalColors());
```

statischen Eigenschaften. Analog zu statischen Methoden werden statische Eigenschaften direkt auf einer Klasse definiert und man benötigt keine Objektinstanz, um auf diese Eigenschaften zuzugreifen. In der Klassensyntax ist es nicht möglich, statische Eigenschaften explizit innerhalb des Klassenkörpers zu definieren. Stattdessen werden statische Eigenschaften als Eigenschaften der Klasse definiert. Eine gebräuchliche Konvention ist es hierbei, die Namen der statischen Eigenschaften in Großbuchstaben zu schreiben:

```
Animal.ANIMAL_COLORS = {
  WHITE: 'Weiß',
  BLACK: 'Schwarz',
  BROWN: 'Braun' }
console.log(Animal.ANIMAL_COLORS);
let bird = new Animal('Birdie',
Animal.ANIMAL_COLORS.BLACK, 5);
console.log(bird.name);
console.log(bird.color);
console.log(bird.age);
```

Da es sich bei der Klassensyntax um eine Vereinfachung der pseudoklassischen Objektorientierung handelt, gilt hier Ähnliches wie dort, wobei die Umsetzung für den Entwickler aber einfacher ist: Statt in Konstruktorfunktionen fasst man abstraktes Verhalten hier in Klassen zusammen, sodass Objektinstanzen der jeweiligen Klasse das Verhalten erben.

Auch die Definition von Vererbungsbeziehungen ist über das Schlüsselwort *extends* einfacher als bei der pseudoklassischen Objektorientierung. Für die Kapselung von Daten gibt es dagegen bis auf die zuvor bei der prototypischen und pseudoklassischen Objektorientierung genannten Möglichkeiten keine speziellen nativen Wege, um etwa die Sichtbarkeit von Eigenschaften und Methoden einzuschränken.

Fazit

JavaScript ist eine objektorientierte, aber keine klassenbasierte Programmiersprache. Daher ist die prototypische Objektorientierung der Sprache am natürlichsten. Die pseudoklassische Objektorientierung tut so, als wäre JavaScript eine klassenbasierte Programmiersprache. Die mit ES6/ES2015 eingeführte Klassensyntax stellt eine syntaktische Vereinfachung für die pseudoklassische Programmierung dar. ■

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

DB Systel stellt die Weichen für den IT-Arbeitsplatz der Zukunft

Lesen Sie die aktuelle Case Study über den Einsatz von Citrix-Technologien

GRATIS Whitepaper

Lesen Sie im kostenlosen Whitepaper von CITRIX®

- Effizienteres Management von IT-Arbeitsplätzen
- Einfachere Migration auf neue Betriebssysteme
- Endgeräteunabhängiger Zugriff auf Applikationen
- Schnellere Reaktionen auf neue Business-Anforderungen
- Neue Möglichkeiten für mobile Zugangsszenarien

Jetzt kostenlos downloaden:
http://digital.internetworld.de/citrix-wp-dbsystel-wump/

Ein Service von:
INTERNET WORLD Business
Ihr Whitepaper bei uns?
Anfragen an sales@nmg.de
oder Tel. 089 / 7 41 17 – 124.

DATENVISUALISIERUNG MIT D3

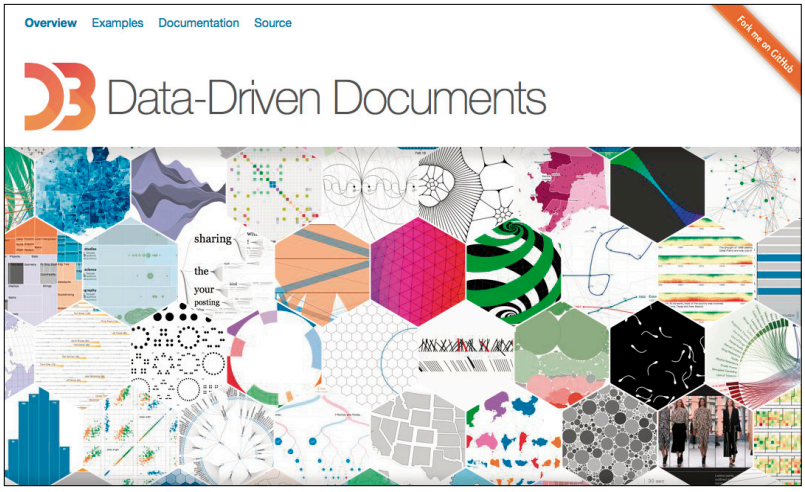
Data-Driven Documents

D3 ist eine mächtige JavaScript-Bibliothek zur Visualisierung großer Datenmengen.

D3 steht für Data-Driven Documents und ist eine mächtige JavaScript-Bibliothek zur Datenvisualisierung. Sie steht unter der BSD-Lizenz. Damit kann man von einfachen Datentabellen über aufwendig gestylte und animierte Charts bis hin zu hochkomplexen und auch interaktiven Diagrammen alles visualisieren.

Die Galerie https://github.com/mbostock/d3/wiki/Gallery gibt einen Überblick über die eindrucksvollen Möglichkeiten der Bibliothek. Klickt man auf eines der Beispiele, erhält man den zugehörigen Programmcode.

Die Bibliothek erlaubt die Manipulation des DOM von HTML-Dokumenten auf der Grundlage von Datensätzen. Dafür nutzt sie die Möglichkeiten der modernen Browser wie Safari, Chrome, Firefox, Opera oder IE9+ ebenso wie die der aktuellen Webstandards CSS3, HTML5 und SVG (Bild 1).



D3-Website: Ein Fundgrube in Sachen Datenvisualisierung für Entwickler (Bild 1)

Funktionsweise

Dabei sind Data-Driven Documents so zu verstehen: Die Daten *(data)* werden vom Anwender geliefert, das Dokument *(documents)* ist ein webbasiertes Dokument, und zwar alles, was ein Webbrowser rendern kann (also HTML, SVG, …). D3 wiederum verbindet die Daten mit dem Dokument. Dabei agiert D3 in den folgenden Bereichen:

- **Loading:** Laden der Daten in den Speicher des Browsers.
- **Binding:** Verbinden der Daten zu Elementen in einem Dokument, und gegebenenfalls die Neuerstellung von Elementen, falls benötigt.
- **Transforming:** Transformation dieser Elemente und entsprechendes Setzen der Eigenschaften.
- **Transitioning:** Überführen von Elementen zwischen den Status aufgrund von User-Eingaben.

Alles, was Sie brauchen, ist ein einfacher Webserver. Legen Sie im Dokument-Root einen Projektordner an und laden Sie dort hinein die neueste Version von D3, die Sie unter https://d3js.org finden. Zum Veröffentlichungszeitpunkt war dies die Version 3.5.16. Wenn Sie eine andere Version benötigen, werden Sie unter https://github.com/mbostock/d3/releases fündig. Das nächste Major-Release finden Sie unter https://github.com/mbostock/d3/tree/4.

Starten wir nun mit einer einfachen HTML-Datei, die wir unter *index.html* im Projektordner ablegen. Dabei gehen wir davon aus, dass sich D3 im Unterordner *d3* befindet. Die Einbindung der Bibliothek sieht so aus:

```
<link href="css/d3.css" rel="stylesheet">
<script type="text/javascript" src="d3/d3.js"></script>
```

Nun legen wir noch ein Verzeichnis *css* an, und dort hinein legen wir eine Datei *d3.css* mit dem folgenden Inhalt:

```
.h-bar {
  min-height: 15px;
  min-width: 10px;
  background-color: steelblue;
  margin-bottom: 2px;
  font-size: 11px;
  color: #f0f8ff;
  text-align: right;
  padding-right: 2px;
}
```

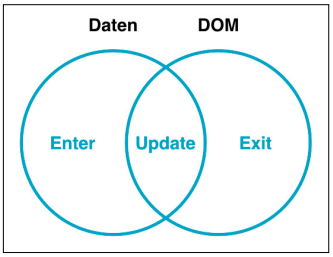
Über den Inspektor können wir prüfen, ob die Bibliothek ordnungsgemäß geladen wurde. Dann schreiben wir unsere erste D3-Anweisung. Wir notieren sie in unserer HTML-Datei:

```
<p id="target"></p>
<script type="text/javascript">
  d3
  .select("p#target")
  .text("Hello world!");
</script>
```

Als Ausgabe erhalten wir *Hello world!*. Das *d3.select*-Kommando wird verwendet, um in D3 ein einzelnes Element aus-

zuwählen. Als Argument wird hierzu (ähnlich wie bei jQuery) ein CSS3-Selektor oder eine Referenz zu einem Element übergeben.

Die Rückgabe ist ein D3-Selektionsobjekt, auf dem weitere Operationen durchgeführt werden können. Unser Beispiel wird daher wie folgt abgearbeitet: Wir haben einen Absatz, dem wir die ID *target* vergeben haben, um diesen später besser ansprechen zu können. Wir selektieren nun den obigen Absatz und setzen den Inhalt auf *Hello world!*.



Data Join: Zurückgeliefert werden drei neue Selections (Bild 2)

D3-Methoden

Es gibt zahlreiche Methoden, um das Ergebnis-Set zu manipulieren:

- **attr(foo,bar):** Spricht das Attribut *foo* an und setzt es auf *bar*. Lässt man den zweiten Parameter weg, so erhält man den Wert des Attributs zurück.
- **classed(foo,bar):** Spricht die Klasse an. Ist *bar = TRUE*, so wird sie gesetzt, ansonsten wird die Klasse zurückgegeben.
- **style(foo,bar):** Setzt den CSS-Stil *foo* (zum Beispiel *font-size)* auf den Wert *bar*.
- **text(foo):** Setzt den Textinhalt eines Containers auf *foo* oder gibt den Text schlicht zurück.
- **html(foo):** Setzt den HTML-Inhalt eines Containers auf *foo* oder gibt den Inhalt zurück.

Es ist zudem möglich, mehrere Elemente zu selektieren. Dafür wird die Methode *selectAll()* verwendet:

```
<div></div>
<div></div>
<div></div>
<script type="text/javascript">
  d3.selectAll("div")
    .attr("class", "blue box");
</script>
```

Hier werden alle * *-Container selektiert und mit der Klasse *blue box* versehen. Über *each()* kann nun auf jedes der vorher selektierten Elemente per Funktion zugegriffen werden:

```
<div></div>
<div></div>
<div></div>
<script type="text/javascript">
  d3.selectAll("div")
    .attr("class", "red box")
    .each(function (d, i) {
       d3.select(this).append("h1").
       text(i);
     });
</script>
```

Neben der Selektion und dem Ausstatten mit einer Klasse wird nun per *each()*-Schleife auf jedes dieser selektierten Elemente einzeln zugegriffen. Der Zugriff selbst erfolgt per *this*.

Über die Funktion *append()* wird zudem ein *<h1>*-Tag ausgegeben, und dort hinein wird die Nummer des Index geschrieben.

Enter-Update-Exit-Pattern

D3 arbeitet in Bezug auf Daten mit einem sogenannten Enter-Update-Exit-Pattern. D3 bindet Daten an das DOM, indem man einer Selection über den Data-Operator die Daten in Form eines Arrays übergibt.

Hier kommt der sogenannte Data-Join zum Einsatz. Zurückgeliefert werden drei neue Selections, die in D3 als *Enter, Update* und *Exit* bezeichnet werden (Bild 2).

In der *Update*-Selection sind die Elemente des DOM mit den Elementen des Data-Arrays in der Reihenfolge ihres jeweiligen Erscheinens verknüpft. Wie man unschwer errät, kann jedoch eine unterschiedliche Anzahl von Elementen im DOM und in den Daten vorhanden sein.

Wenn im DOM mehr Elemente vorhanden sind als Daten, dann liegen diese Elemente in der *Exit*-Selection, mit der man die Elemente aus dem DOM entfernen kann.

Gibt es mehr Daten als DOM-Elemente, dann befinden sich diese Daten in der *Enter*-Selection, mit der die fehlenden Elemente im DOM erstellt werden können:

```
<p style="color:green;">Ein bereits existierendes
Element</p>
<script>
  var body = d3.select('body');
  var p = body.selectAll('p').data([1,2,3]).
  style('color', 'red');
  p.enter().append('p').text( function(d){
  return 'Neues Element aus Daten ' + d; });
  p.exit().remove();
</script>
```

Ein bereits existierendes Element

Neues Element aus Daten 2

Neues Element aus Daten 3

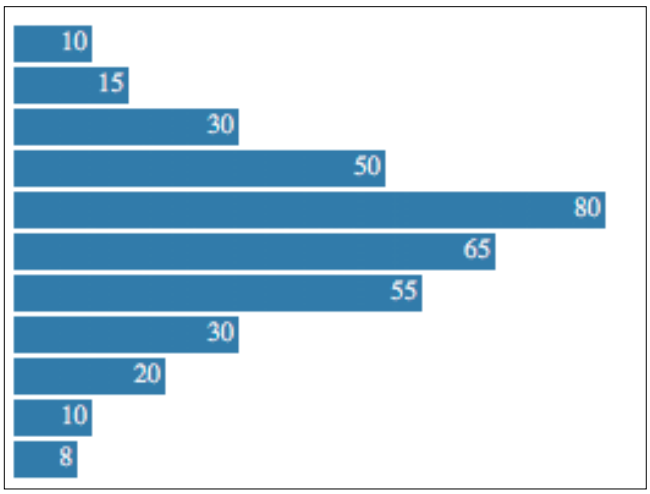
Weitere Elemente in der *enter*-Funktion (Bild 3)

Im Beispiel sieht man einen Absatz, der grün eingefärbt wurde. Nun wird der Body selektiert und in diesem alle Absätze. Darauf wird die *data()*-Funktion angewendet und alle Absätze rot eingefärbt. Da es aber nur einen Absatz, dafür aber drei Daten-Einträge gibt, legen wir in der *enter*-Funktion für die restlichen Daten weitere Elemente an (Bild 3).

Wir wollen nun Daten mit Hilfe eines Balkendiagramms visualisieren. Dazu erstellt man zunächst einen Datensatz, der aus zehn Daten besteht. Diese werden über die Methode ▶

render() visualisiert. Nun selektieren wir in der *Enter*-Sektion all jene * *-Tags, die eine Klasse *h-bar* besitzen, und mappen darauf die Daten. Tatsächlich ist unser Markup aber absolut leer – das heißt, es existieren überhaupt keine Tags.

Genau das macht sich aber die *Enter*-Funktion zunutze – das heißt, gemäß dem Enter-Update-Exit-Paradigma werden nun alle Elemente angelegt, die es im DOM noch nicht gibt, und zudem ein **-Tag mit einem Label erzeugt. In der *Update*-Funktion wird nun gemäß den Daten ein Attribut *width* gesetzt und in das Label die Breite eingetragen. Damit wird später eine Visualisierung (vertikale Balken) möglich gemacht. Über die *setInterval()*-Funktion wird nun alle 1,5 Sekunden der erste Wert aus dem Data-Array entfernt und ein neuer Wert per Zufall in das Data-Array gepusht. Die neue Liste wird anschließend als Balkendiagramm ausgegeben (Bild 4). Das komplette Listing steht unter www.webundmobile.de zur Verfügung.



Liste als Balkendiagramm ausgeben (Bild 4)

Skalen, Achsen und Übergänge

Im nächsten Beispiel wollen wir uns sowohl die Skalen ansehen als auch die Achsen-Erzeugung und Übergänge zwischen Visualierungen. Dafür erzeugen wir ein Koordinatensystem mit einer x- und einer y-Achse und platzieren darauf 50 zufällig angeordnete Datenpunkte. Hier verwenden wir SVG als Träger der Visualisierung. Aber zuerst müssen wir noch etwas CSS zu unserer Datei *d3.css* hinzufügen, um die Achsen samt Beschriftung zu stylen:

```
.axis path,
.axis line {
  fill: none;
  stroke: black;
  shape-rendering: crispEdges;
}
.axis text {
  font-family: sans-serif;
  font-size: 11px;
}
```

Nun bauen wir das entsprechende HTML auf. Ganz oben befindet sich ein Absatz mit einem Text, der signalisiert, dass bei Klick darauf sich die Visualisierung neu aufbaut. Anfangs legen wir fest, dass die Visualisierung immer 500 Pixel Breite und 300 Pixel Höhe haben soll. Später passen wir die Achsen exakt auf diese Maße an. Nun bauen wir das Datenset als Array mit Hilfe der Funktion *randomValue()* auf. Sie sorgt dafür, dass 50 zufällige Punkte entstehen, die wir darstellen wollen. Dann werden die Achsen x und y aufgebaut. Dafür verwenden wir die Funktion *d3.scale.linear()*. Über *domain()* wird festgelegt, dass die Achse Werte von *0* bis zum maximalen Wert behandeln und diese per *range()* innerhalb der Größe der festgelegten Werte inklusive Padding ausgeben soll:

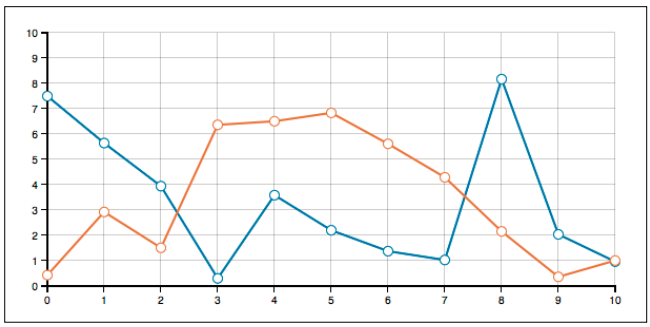
```
var w = 500;
var h = 300;
var padding = 30;
var dataset = randomValue();
var xScale = d3.scale.linear()
.domain([0, d3.max(dataset, function(d) { return d[0];
})])
.range([padding, w - padding * 2]);
var yScale = d3.scale.linear()
.domain([0, d3.max(dataset, function(d) { return d[1];
})])
.range([h - padding, padding]);
```

Nun werden die Achsen als SVG-Objekte gerendert. Dabei wird über *scale()* das Achsen-Objekt verwendet, als Orientierung (Funktion *orient()*) wird für die x-Achse unten und für die y-Achse links festgelegt. Zudem werden fünf Unterteilungen in der Scale über *ticks()* eingebracht:

```
var xAxis = d3.svg.axis()
  .scale(xScale)
  .orient("bottom")
  .ticks(5);
var yAxis = d3.svg.axis()
  .scale(yScale)
  .orient("left")
  .ticks(5);
```

Anschließend wird das SVG-Objekt selbst erstellt und als *<svg>*-Tag mit der festgelegten Breite und Höhe im DOM eingefügt. Schließlich werden die Datenpunkte als Kreise mit einem Radius von 2 px im Koordinatensystem platziert:

```
var svg = d3.select("body")
  .append("svg")
  .attr("width", w)
  .attr("height", h);
svg.selectAll("circle")
  .data(dataset)
  .enter()
  .append("circle")
  .attr("cx", function(d) {
   return xScale(d[0]); })
```



Charts: Es können verschiedene Chart-Typen generiert werden (Bild 5)

```
  .attr("cy", function(d) {
   return yScale(d[1]); })
  .attr("r", 2);
```

Damit bei Klick auf den Absatz beim Neuaufbau der Visualisierung eine Animation zwischen altem und neuem Zustand erfolgt, bringt man entsprechende CSS3-Anweisungen ein:

```
svg.append("g") // <-H
.attr("class", "x axis")
.attr("transform", "translate(0," + (h - padding) + ")")
.call(xAxis);
svg.append("g")
.attr("class", "y axis")
.attr("transform", "translate(" + padding + ",0)")
.call(yAxis);
```

Der Klick auf den Absatz wird per *on()*-Funktion getriggert. Anschließend wird das Datenset neu erstellt (per Zufallsfunktion) und die Achsen werden entsprechend neu skaliert.

Nun werden alle Datenpunkte selektiert und diese per Übergang über *transition()* auf die neuen Datenpunkte mit einer Animationslänge per *duration()* von 1000 ms animiert. Zudem werden die Punkte während der Animation kurzzeitig mit der Farbe Magenta eingefärbt und größer dargestellt:

```
d3.select("p").on("click", function() { // <-I
dataset = randomValue();
xScale.domain([0, d3.max(dataset, function(d) { return
d[0]; })]); // <-J
yScale.domain([0, d3.max(dataset, function(d) { return
d[1]; })]);
  svg.selectAll("circle") // <-K
    .data(dataset)
    .transition()
    .duration(1000)
    .each("start", function() {
      d3.select(this)
        .attr("fill", "magenta")
        .attr("r", 7); })
        .attr("cx", function(d) {
         return xScale(d[0]); })
        .attr("cy", function(d) {
          return yScale(d[1]); })
        .transition()
        .duration(1000)
        .attr("fill", "black")
        .attr("r", 2);
```

Natürlich müssen auch die neuen Achsen dargestellt werden – in diesem Fall ebenfalls animiert:

```
svg.select(".x.axis")
.transition()
.duration(1000)
.call(xAxis);
svg.select(".y.axis")
.transition()
.duration(1000)
.call(yAxis);
```

Die nachfolgende Funktion erzeugt das zufällige Datenset. Dabei werden 50 Datenpunkte mit einem Wert zwischen 0 und 1000 ermittelt und als Array zurückgegeben:

```
function randomValue() {
  dataset = [];
  var numDataPoints = 50;
  var maxRange = Math.random() * 1000;
  for (var i = 0; i < numDataPoints; i++) {
    var newNumber1 = Math.floor(Math.random() *
    maxRange);
    var newNumber2 = Math.floor(Math.random() *
    maxRange);
      dataset.push([newNumber1, newNumber2]);
    }
    return dataset; }
```

Mit dem bisherigen Wissen können nun beliebige Chart-Typen wie Tortendiagramme, Bubble-Charts, Ring-Charts, Stack-Layouts, Network-Layouts, Tree-Maps und Layered Area Charts realisiert werden (Bild 5).

Fazit

Derzeit gibt es sicherlich kein besseres Werkzeug, um selbst komplexe Daten in unzähligen Darstellungsformen zu visualisieren. Zudem existieren zahlreiche Integrationen, zum Beispiel in AngularJS, React oder gar R, sodass D3 sicherlich den Standard in der Datenvisualisierung setzt. ■

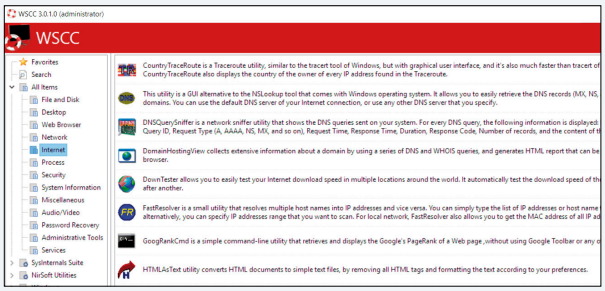
Patrick Lobacher
ist Digital Native, Entwickler, Berater, Coach und Autor zahlreicher Fachbücher und Fachartikel. Er ist Vorstandsvorsitzender der Pluswerk AG, die digitale Kommunikationslösungen konzipiert, umsetzt und betreut.
www.pluswerk.ag

ÜBERBLICK

CD-Highlights

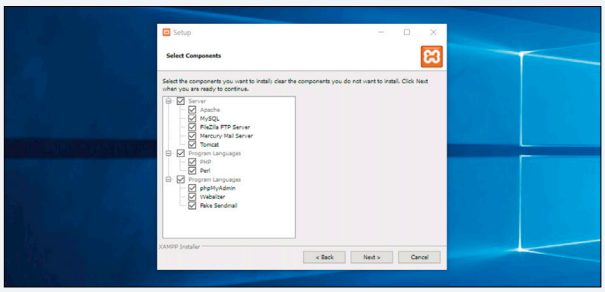
Auf der Heft-CD finden Sie Werkzeuge für Webentwickler und nützliche Tools.

Windows System Control Center 3.0.1



Das Windows System Control Center – kurz WSCC – ist eine umfangreiche Tool-Sammlung, die vor allem für versierte PC-Anwender, Netzwerkprofis und Programmierer interessant ist. Mit dem WSCC haben Sie Zugriff auf über 300 System-Tools aus verschiedenen Bereichen. Das Hauptmenü von WSCC ist übersichtlich gestaltet, sodass Sie sich leicht zurechtfinden. In der linken Spalte sind die Tools nach Herstellern in die Rubriken Sysinternals Suite, NirSoft Utilities und Windows sowie in zahlreiche Unterrubriken eingeteilt. Gutes Extra: Ein integrierter Update-Manager hält die Programme stets aktuell und spielt neue Tools nach Verfügbarkeit ein.

XAMPP 7.0.6



Ein Apache-Webserver installiert sich nicht so leicht – an der ein oder anderen Stelle sind Systemkenntnisse zum Einspielen der Pakete erforderlich. Noch schwieriger wird es, wenn weitere Pakete wie MySQL, PHP oder Perl dazukommen. Wesentlich einfacher und schneller geht's mit XAMPP: Das Paket enthält alle wichtigen Module und installiert diese auf Knopfdruck. Nach dem Start funktioniert alles wie gewünscht, und Sie können mit der Konfiguration der Module fortfahren.

Weitere Freeware-Highlights

7-Zip 16.00

Die Open-Source-Software entpackt unter anderem Archive der Formate ARJ, CAB, ISO, LZH, LZMA, MSI, NSIS, RAR, VHD und WIM. In den Formaten 7z, GZIP, TAR und ZIP kann 7-Zip Dateien sowohl komprimieren als auch dekomprimieren. Besonders hohe Kompressionsraten lassen sich mit dem Algorithmus 7z erreichen.

ISO Workshop 7.0

Das Programm stellt Abbilder von CDs, DVDs und Blu-rays her. Außerdem durchsucht und konvertiert das Tool solche Images und brennt daraus wieder eine lauffähige Scheibe. ISO Workshop liest die Formate ISO, CUE, BIN, NRG, CDI, MDF, IMG, DMG und B6I, Beim Brennen und Schreiben werden ISO und BIN/CUE unterstützt.

WhatsApp für Windows 0.2.684

Der bekannte WhatsApp-Messenger ist jetzt auch als native App für Windows-PCs erhältlich. Wie beim Webclient ist eine Anmeldung via QR-Code-Scan auf dem Smartphone (iOS, Android, Windows Mobile) erforderlich. Der Scan lässt sich in den Einstellungen der Smartphone-App unter dem Menüpunkt *WhatsApp Web* starten.

CarotDAV 1.12.8

Mit der Client-Software CarotDAV greifen Sie per WebDAV-Protokoll auf Online-Speicher zu. Das Tool ermöglicht auch FTP-Zugriffe und spricht die Cloud-Speicher von Box, Dropbox, Google Drive, Skydrive und Sugarsync direkt an. Der Vorteil ist, dass Sie auf diese Weise viele Cloud-Dienste unter einen Hut bringen können.

Filezilla Server 0.9.57

Mit Filezilla Server erstellen Sie auf Ihrem Windows-PC einen eigenen FTP-Server. Nach dem ersten Start des Programms fügen Sie Benutzer hinzu und geben Verzeichnisse frei. Unter *Files* und *Directories* legen Sie fest, ob die Benutzer Dateien und Verzeichnisse schreiben, lesen oder löschen dürfen.

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

WEB STORAGE API

Browserspeicher

Ein Blick auf das Web Storage API und die dafür relevanten JavaScript-Bibliotheken.

Das Web Storage API (https://www.w3.org/TR/webstorage) ist am 19. April 2016 vom W3C in den Status »Recommendation« erhoben worden und wird von allen namhaften Browsern unterstützt (Bild 1). Eine gute Gelegenheit also, das API noch einmal genauer unter die Lupe zu nehmen.

Über das Web Storage API können Webentwickler innerhalb einer Webanwendung Daten lokal auf Clientseite speichern. Dabei unterscheidet man prinzipiell zwischen zwei verschiedenen Arten von Browserspeichern: dem lokalen Browserspeicher, der übergreifend über verschiedene Browserfenster und -tabs zugänglich ist (Bild 2), sowie dem Sessionspeicher, der nur innerhalb eines einzelnen Browsertabs verfügbar ist (Bild 3).

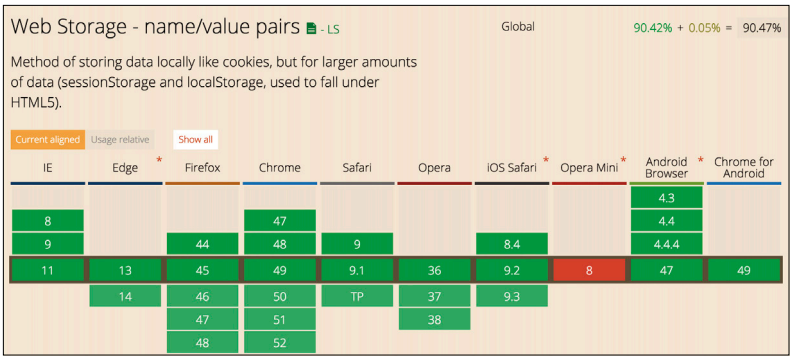
Auf Ersteren, das heißt auf den lokalen Speicher, kann über das globale Objekt *localStorage* zugegriffen werden, auf Letzteren, den Sessionspeicher, über das ebenfalls globale Objekt *sessionStorage*.

Bezüglich des API sind beide Browserspeicher identisch: Beide implementieren das Storage-Interface und verfügen somit über die gleichen Methoden und Eigenschaften (Tabelle 1).

Werte speichern

Das Speichern von Werten (beziehungsweise Schlüssel-Wert-Paaren) in einen der Browserspeicher geschieht über die Methode *setItem()*. Ihr übergibt man als ersten Parameter einen Schlüssel, unter dem der entsprechende Wert abgespeichert werden soll, sowie als zweiten Parameter den zu speichernden Wert. Dies funktioniert für den lokalen Browserspeicher genauso wie für den Session-Browserspeicher:

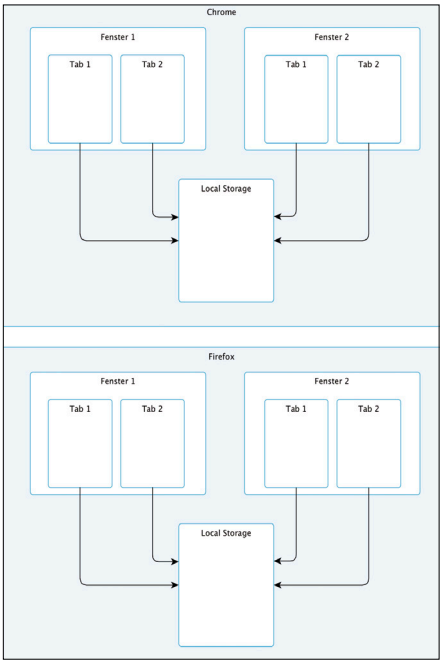
```
// Speichern von Werten in den
// lokalen Browserspeicher
localStorage.setItem
('firstname', 'Max');
localStorage.setItem
('lastname', 'Mustermann');

// Speichern von Werten in den
// Session-Browserspeicher
sessionStorage.setItem
('firstname', 'Max');
sessionStorage.setItem
('lastname', 'Mustermann');
```



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

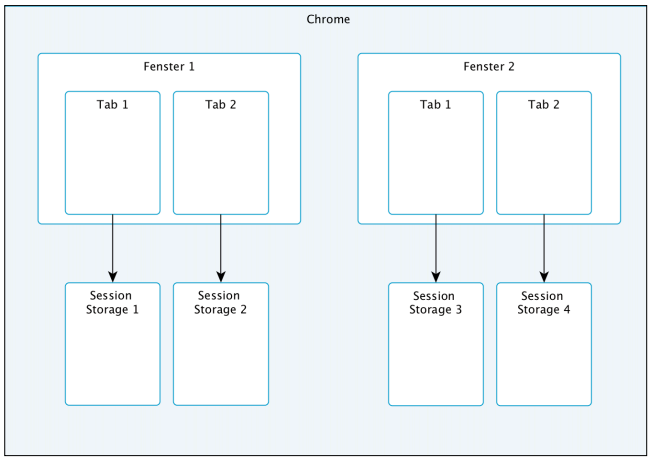
Bei dem zu speichernden Wert darf es sich in beiden Fällen ausschließlich um eine Zeichenkette handeln. Das bedeutet, dass Objekte zunächst per *JSON.stringify()* in eine solche umgewandelt werden müssen:

```
let user = {
  firstname: 'Max',
  lastname: 'Mustermann'
}
// Falsch: die Zeichenkette [object
// Object] wird gespeichert
localStorage.setItem(
  'user',
  user
);
// Richtig: das Objekt wird im
// JSON-Format gespeichert
localStorage.setItem(
  'user',
  JSON.stringify(user)
);
```



Das Prinzip des lokalen Browserspeichers (Bild 2)

Macht man dies nicht, wird der Wert verwendet, der von der *toString()*-Me-



Das Prinzip des Sessionspeichers (Bild 3)

thode des jeweiligen Objekts zurückgegeben wird, was standardmäßig die Zeichenkette *[object Object]* ist. Nicht wirklich das, was man möchte.

Analog zur Methode *setItem()* für das Speichern von Werten in den Speicher gibt es die Methode *getItem()*, um Werte aus dem Speicher anhand ihres Schlüssels zu lesen:

```
let firstname = localStorage.getItem('firstname');
let lastname = localStorage.getItem('lastname');
console.log(firstname);
console.log(lastname);
```

Beim Auslesen von zuvor gespeicherten Objekten muss man natürlich den entsprechenden Wert wieder per *JSON.parse()* in ein Objekt umwandeln. Um beispielsweise das zuvor gespeicherte *user*-Objekt wieder aus dem Speicher zu lesen, geht man so vor:

```
let user = localStorage.getItem('user');
user = JSON.parse(user);
console.log(user.firstname);
console.log(user.lastname);
```

Für das Aktualisieren von Werten in einem der Browserspeicher gibt es dagegen keine separate Methode: Hier kommt wieder die Methode *setItem()* zum Einsatz. Diese sorgt nämlich auch dafür, dass ein Wert, falls dieser beziehungsweise der entsprechende Schlüssel bereits im Speicher existiert, einfach überschrieben wird:

```
localStorage.setItem('firstname', 'Max');
localStorage.setItem('lastname', 'Mustermann');
let firstname = localStorage.getItem('firstname');
let lastname = localStorage.getItem('lastname');
console.log(firstname);
console.log(lastname);
localStorage.setItem('firstname', 'Moritz');
localStorage.setItem('lastname', 'Peterson');
firstname = localStorage.getItem('firstname');
lastname = localStorage.getItem('lastname');
console.log(firstname);
console.log(lastname);
```

Über die Methode *removeItem()* lassen sich gezielt einzelne Einträge aus dem Browserspeicher löschen, wobei als Parameter hier der Schlüssel des entsprechenden Eintrags zu übergeben ist. Möchte man hingegen nicht nur einen Eintrag, sondern alle Einträge eines Browserspeichers löschen, bedient man sich der Methode *clear()*:

```
localStorage.setItem('firstname', 'Max');
localStorage.setItem('lastname', 'Mustermann');
let firstname = localStorage.getItem('firstname');
let lastname = localStorage.getItem('lastname');
console.log(firstname);
console.log(lastname);
localStorage.removeItem('firstname');
localStorage.removeItem('lastname', 'Peterson');
```

Um auf Änderungen im Browserspeicher zu reagieren, können Sie über die Methode *addEventListener()* des *window*-Objekts einen Event Listener für das *storage*-Event registrieren. Dieses Event wird immer dann ausgelöst, wenn sich der Browserspeicher ändert.

Das entsprechende Event-Objekt, das dem Event Listener als Argument übergeben wird, ist vom Typ *StorageEvent* und enthält neben den standardmäßig für *Event*-Objekte vorhandenen Eigenschaften noch weitere:

```
window.addEventListener('storage', (e) => {
  console.log(e.key);
  console.log(e.oldValue);
  console.log(e.newValue);
```

▶

Tabelle 1: Eigenschaften und Methoden von Storage

Eigenschaft/ Methode	Typ	Beschreibung
length	Number	Enthält die Anzahl der Elemente in dem Storage.
clear()	void	Entfernt alle Schlüssel-Wert-Paare aus dem Storage.
getItem (String key)	String	Liefert für einen Schlüssel den entsprechenden Wert aus dem Storage.
key(long index)	String	Liefert den Namen des n-ten Schlüssels aus dem Storage.
remove Item(String key)	void	Entfernt – falls vorhanden – das Schlüssel-Wert-Paar für den übergebenen Schlüssel aus dem Storage.
setItem (String key, String value)	void	Fügt für den entsprechenden Schlüssel einen Wert in den Storage oder – für den Fall, dass bereits ein gleichnamiger Schlüssel existiert – überschreibt den mit dem Schlüssel assoziierten Wert.

Tabelle 2: Die Eigenschaften von StorageEvent

Eigenschaft	Typ	Beschreibung
key	String	Enthält den Schlüssel des aktualisierten Eintrags.
oldValue	String	Enthält den alten Wert, der hinter dem Schlüssel hinterlegt war.
newValue	String	Enthält den neuen Wert, der nun hinter dem Schlüssel hinterlegt ist.
url	String	Enthält den URL des Dokuments, für das ein Eintrag geändert wurde.
storageArea	Storage	Enthält eine Referenz auf das Storage-Objekt, das geändert wurde.

```
console.log(e.url);
console.log(e.storageArea); });
```

Die Eigenschaft *key* enthält den Schlüssel, für den sich der Eintrag geändert hat, *oldValue* enthält den alten Wert, *newValue* den neuen Wert, die Eigenschaft *url* enthält den URL des Dokuments, in dem der Eintrag geändert wurde, und *storageArea* enthält das Storage-Objekt, das von der Änderung betroffen ist (Tabelle 2).

Das Web Storage API definiert ein recht übersichtliches API und ist bezüglich seiner Funktionalität klein, aber zweckmäßig gehalten. Mit der Zeit hat die JavaScript-Community allerdings eine Reihe von Bibliotheken entwickelt, die sich Aspekten wie Verschlüsselung, Komprimierung, Speichern von sehr vielen Daten, Speichern von verschiedenen Datentypen und einiges mehr annehmen (Tabelle 3).

Fazit

Über das Web Storage API lassen sich Daten in Form von Schlüssel-Wert-Paaren in einem clientseitigen Browserspeicher ablegen, wobei zwischem lokalem Speicher und Sessionspeicher unterschieden wird. Da als Werte allerdings nur Zeichenketten erlaubt sind, müssen JavaScript-Objekte entsprechend beim Speichern über *JSON.stringify()* und beim Lesen über *JSON.parse()* umgewandelt werden. Das API wird mittlerweile von allen gängigen Browsern unterstützt; zudem steht eine große Auswahl an Bibliotheken zur Verfügung, die das API um verschiedene Features erweitern. ■

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

Tabelle 3: JavaScript-Bibliotheken zum Web Storage API

Bibliothek	URL	Beschreibung
Lockr	https://github.com/tsironis/lockr	Wrapper für das Web Storage API, welches das Speichern von Objekten, Arrays, Zahlen und anderen Datentypen ermöglicht und dabei ein Redis-ähnliches API zur Verfügung stellt.
Barn	https://github.com/arokor/barn	Ebenfalls ein Wrapper um das Web Storage API, das ein Redis-ähnliches API zur Verfügung stellt.
Store.js	https://github.com/marcuswestin/store.js	Bibliothek, die für ältere Versionen des Internet Explorers auf *userData* zurückgreift.
lsCache	https://github.com/pamelafox/lscache	Bibliothek, die *memcache*-Funktionalität nachbildet und damit erlaubt, einen Zeitstempel anzugeben, ab dem der jeweils gespeicherte Eintrag im Browserspeicher gelöscht wird.
secStore.js	https://github.com/jas-/secStore.js	Wrapper um das Web Storage API, welches die Daten unter Verwendung von SJCL (Stanford Javascript Crypto Libraries, http://bitwiseshiftleft.github.io/sjcl) verschlüsselt
localForage	https://github.com/mozilla/localForage	Wrapper um das Web Storage API, IndexedDB und WebSQL, der eine einheitliche Schnittstelle für alle genannten Persistenztechniken anbietet.
Basil.js	https://github.com/Wisembly/basil.js	Bibliothek, die dem Web Storage API eine zusätzliche Abstraktionsebene hinzufügt.
lz-string	https://github.com/pieroxy/lz-string	Bibliothek, über die Daten komprimiert im Browserspeicher persistiert werden können.
alaSQL.js	https://github.com/agershun/alasql	SQL-Datenbank, die im Browser, unter Node.js und unter Apache Cordova verwendet werden kann und dabei auch mit einem unterliegenden Browserspeicher umgehen kann.
LargeLocal Storage	https://github.com/tantaman/LargeLocalStorage	Bibliothek, die das Speichern von sehr vielen Daten im Browserspeicher ermöglicht.
LokiJS	https://github.com/techfort/LokiJS/	Dokumentorientierte Datenbank auf Basis des Browserspeichers.

DDC .NET Developer Conference 2016

- **05.12.2016 – DevSessions:**
  2 halbtägige Workshops
- **06.12.2016 – Konferenz:**
  1 Keynote, 2 Themenstränge,
  12 Vorträge
- **07.12.2016 – Workshops**
  Ganztägiges Praxistraining

Köln, Pullman Cologne

Trends, Lösungen und Know-how für für Profi-Entwickler

Themenauswahl:

- **Softwarequalität:**
  Unit Testing, DDD, Refactor
- **Core:** .NET Core, Strings,
  Async/Await
- **Frontend:** UI Engineering,
  WPF, ASP.NET Core
- **Any App:** UWP, Xamarin,
  Microservices

inklusive
SMART DATA
Developer Conference
Big Data & Smart Analytics

dotnet-developer-conference.de **#netdc16** **DDConference**

Veranstalter: developer media — Neue Mediengesellschaft Ulm mbH

Präsentiert von: dotnetpro

DDC .NET Developer Conference 2015

DevSessions: 05. Dezember 2016

	Frontend	Frontend	Softwarequalität	Architektur
09.00 – 13.00*	MVVM-Pattern mit WPF *Bernd Marquardt* Level 200	UWP Apps *André Krämer* Level 100	Software verbessern - mit aim42 *Gernot Starke* Level 100	Mikroservice Architektur mit Azure Service Bus *Torsten Helmich* Level 200
13.00 – 14.00	Mittagspause			
	Frontend	**Frontend**	**Softwarequalität**	**Backend**
14.00 – 18.00*	The WPF First Aid Kit *David Würfel* Level 300	Dreamteam: ASP.NET Core und Angular 2.0 *Johannes Hoppe* Level 100	TDD für Testmuffel *Hendrik Lösch* Level 200	Plattformunabhängiger Datenzugriff mit Entity Framework Core 1.0 *Dr. Holger Schwichtenberg* Level 100

*10.30 – 11.00 und 15.30 – 16.00 Kaffeepause

Konferenz-Programm: 06. Dezember 2016

SMART DATA Developer Conference
Big Data & Smart Analytics

08.45	Begrüßung durch den Veranstalter		Begrüßung
09.00 – 09.55	**Keynote:** Welcome HoloLens • *Damir Dobric und Andreas Erben*		Success Story: Industrie 4.0 & Big Data
	Softwarequalität	**Frontend**	**SMART DATA**
10.00 – 10.55	Fachliche Architektur mit DDD für .NET *Henning Schwentner* Level 200	Windows 10 UI Engineering für WPF-Entwickler *Thomas Immich* Level 300	Datenqualität: Big Data Matching *Dr. Hanna Köpcke*
11.00 – 11.30	Kaffeepause		Kaffeepause
11.30 – 12.25	12 tips for unit tests that don't cripple your codebase *Dennis Doomen* Level 200	Datenbindung Deluxe – Deep Dive in das Binding von WPF *Christian Giesswein* Level 400	4x4: Vier Real-World Beispiele bei vier Cloud Providern *Danny Linden*
12.30 – 13.25	Refactoring C# Legacy Code *Stefan Lieser* Level 300	Running your ASP.NET Core app in a Docker container *Maurice de Beijer* Level 200	NoSQL: Einführung in Graphdatenbanken mit Neo4j *Tobias Trelle*
13.30 – 14.30	Mittagspause		Mittagspause
14.00 – 14.30	Lunchsession		Recht: „German Cloud" *Dobric und Erben*
	Core	**Any App**	**SMART DATA**
14.30 – 15.25	.NET Core *David Tielke* Level 200	UWP - New Horizons *Marek Pohanka und Daniel Bauer* Level 200	Recommender-Algorithmen mit R, Spark DataFrames und Spark MLlib *Henrik Behrens*
15.30 – 16.00	Kaffeepause		Kaffeepause
16.00 – 16.55	Async/Await: Out of Context *Tim Bussmann* Level 200	Xamarin - .NET für die Hosentasche *Sven-Michael Stübe* Level 200	Smart Analytics: Streaming mit Apache Flink *Stephan Papp*
17.00 – 18.00	Die dunkle Wahrheit über Strings im .NET Bereich *Christian Giesswein* Level 400	Microservices and Azure Service Fabric *Johannes C. Dumitru* Level 300	Visual, Active & Mobile Reports nach HICHERT®SUCCESS mit IBM Cognos *Holger Gerhards*
18.30 – 20.30	**Abendveranstaltung** mit Night Coding		

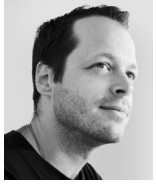
Programmänderung vorbehalten

Ihre Referenten (u.a.)

Daniel Bauer,
Kupferwerk GmbH

Damir Dobric,
DAENET Corporation

Dennis Doomen,
.NET Architect, Coach and Writer



Andreas Erben,
DAENET Corporation

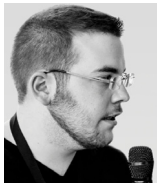
Christian Giesswein,
Giesswein Software-Solutions

Torsten Helmich,
SW Architekt und Scrum Master

Johannes Hoppe,
HAUS HOPPE ITS

Dr. Hanna Köpcke,
Webdata Solutions GmbH

Stefan Lieser,
Mitgründer der Clean Code Developer Initiative



Danny Linden,
OnPage.org

Stefan Papp,
The unbelievable Machine Company GmbH

Gernot Starke,
Gründungsmitglied des iSAQB e.V.

David Tielke,
david-tielke.de

Tobias Trelle,
codecentric AG

Ralf Westphal,
Mitgründer der Clean Code Developer Initiative

Workshops: 07. Dezember 2016

Workshop 1 | Moderne Anwendungsentwicklung mit .NET

David Tielke Uhrzeit: 09.00 – 18.00 Uhr

In diesem Workshop verschafft Ihnen Trainer David Tielke einen Überblick über die verschiedensten Anwendungsarten: moderne Desktopanwendung mit WPF, Datenzugriff mit dem Entity Framework, moderne Webanwendungen mit ASP.NET MVC und WebAPI, Windows 10 Universal Apps, Anwendung mit Windows IoT auf einem Raspberry PI, Anwendung in die Cloud bringen.

Workshop 2 | Architektur Deluxe mit C# und .NET

Christian Giesswein Uhrzeit: 09.00 – 18.00 Uhr

In diesem Workshop geht es darum Merkmale von "gutem" Code zu erläutern und auch die Techniken. Zum Beispiel lernen Sie Dependency Injection und Inversion-Of-Control praxisnah für jedes Projekt kennen, damit auch Ihre Software innerhalb kürzester Zeit eine höhere Qualität aufweist.

Workshop 3 | TDD 2.0 – Refactoring unnötig, offensichtlich, unvermeidbar

Ralf Westphal Uhrzeit: 09.00 – 18.00 Uhr

Lassen Sie sich überraschen, wie "grüner Code" einerseits schon refaktorisiert entstehen kann - oder andererseits die Refaktorisierung unvermeidbar wird. Einführung in "TDD as if you meant it" und "Informed TDD".

Training: 08. Dezember 2016

developer media Training | Softwarequalität

David Tielke Uhrzeit: 09.00 – 18.00 Uhr

Als erfahrener Trainer zeigt Ihnen David Tielke, wie Sie Softwarequalität spielend einfach in Ihren eigenen Projekten unterbringen, was die ersten Schritte sind und wie Schritt-für-Schritt ein mächtiger, nachhaltiger und transparenter Prozess für alle Beteiligten aufgebaut wird. Dabei liegt der Fokus auf allen Arten der Softwarequalität: richtig guten Code schreiben, kontrollieren, Architektur, Toolunterstützung, Testing ...

10% Rabatt bei Kombi-Anmeldung mit DDC

Kooperationspartner (Stand 13.05.2016):

BDOA · digitalbusiness Cloud & IoT · e-commerce magazin Geschäftserfolg im Internet · W&V

DIE WICHTIGSTEN NEUERUNGEN VON SWIFT 2.2

Die richtige Richtung

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 Entwicklern aufgenommen und Apple arbeitete seitdem fleißig an weiteren neuen Features und Verbesserungen der Sprache (Bild 1).

Das ging sogar so weit, dass Apple auf der WWDC 2015 bekannt gab, den Quellcode für den Compiler und die Standard Library als Open Source freizugeben. Diese Bekanntmachung wurde dann Ende letzten Jahres Wirklichkeit, als Apple die Plattform Swift.org startete und darüber nun seitdem die weitere Entwicklung von Swift steuert, Download-Pakete bereitstellt und in einem separaten Blog über die nächsten Schritte und Features informiert (Bild 2).

In eben jenem Blog gab Apple Ende März bekannt, eine neue Version 2.2 von Swift fertiggestellt und veröffentlicht zu haben. Zusammen mit Swift 2.2 erschien zeitgleich auch eine aktualisierte Version von Xcode mit der Versionsnummer 7.3, die komplett kompatibel zum Swift-Update ist und entsprechend aktualisierte Compiler sowie eine neue Standard Library mitbringt.

Dabei ist das Besondere an diesem Release, dass es das erste nach Freigabe des Swift-Quellcodes als Open Source ist und somit die Community ihren Teil zum aktuellsten Stand der Sprache beigetragen hat. Das betont auch Apple in einem entsprechenden Blog-Post von Ende März, in dem sie die wichtigsten neuen Features und Änderungen von Swift 2.2 im Detail vorstellen.

Auf der Worldwide Developers Conference (WWDC) 2014 hat Apple die neue Programmiersprache Swift vorgestellt (Bild 1)

Swift Language Version Build Configuration

Apple ist durchaus bewusst, dass Entwickler, die sich bereits von Beginn an mit Swift beschäftigen und eigene Projekte damit umsetzen, bei neueren Versionen der Sprache meist große Teile ihres Codes entsprechend anpassen müssen. Was für einzelne App-Projekte noch überschaubar und dank Xcode-Migrationsassisent womöglich auch schnell vonstatten gehen kann, ist für die Anbieter von Libraries in der Regel nicht so einfach zu lösen. Schließlich soll eine Library nicht zwingend nur mit der neuesten Version von Swift funktionieren, sondern auch ältere Versionen unterstützen.

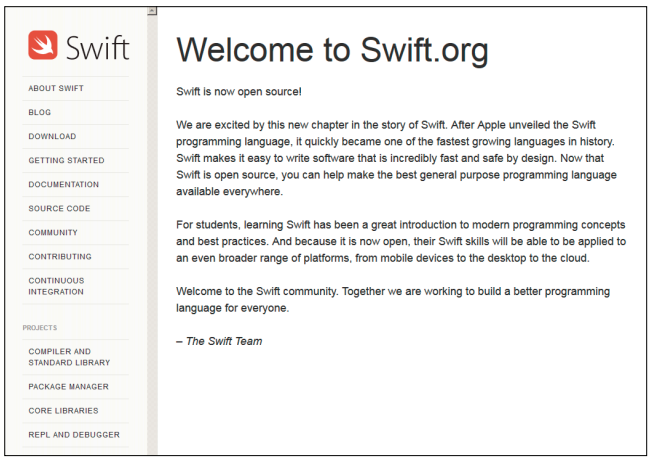
Um diesem Problem gerecht zu werden, mussten Entwickler eigener Swift-Libraries bisher idealerweise immer verschiedene Versionsstände ihrer Library für verschiedene Swift-Versionen vorhalten und separat anbieten. Typischerweise kamen dazu dann verschiedene Repository-Branches zum Einsatz, die eindeutig einer Swift-Version zugeordnet werden konnten, und daraus wurden dann die passenden Libraries generiert.

Mit Swift 2.2 führt Apple aber nun ein neues Feature ein, das diese Problematik aus dem Weg räumen soll. Die Rede ist dabei von der sogenannten Swift Language Version Build Configuration. Dabei handelt es sich um eine neue Direktive, die zur Compile-Time geprüft und ausgeführt wird. Diese erlaubt es, einen Code-Block nur dann auszuführen, wenn er mindestens einer vorgegebenen Swift-Version entspricht:

```
#if swift(>=3.0)
print("Running Swift 3.0 or later")
#else
print("Running Swift 2.2 or earlier")
#endif
```

Die neue Direktive wird mittels *#if swift* eingeleitet. Innerhalb eines anschließenden runden Klammernpaars wird die Swift-Version angegeben, die mindestens vorliegen muss, um den darauffolgenden Code auszuführen. Dabei können mehrere Befehle in mehreren Zeilen aneinandergereiht werden.

Dabei ist die Angabe von >= innerhalb des runden Klammernpaars vor der Swift-Version nicht obligatorisch. Es muss immer implementiert werden, auch gibt es zum jetzigen Zeitpunkt keine Alternativen dazu. Apple behält sich aber vor, langfristig weitere Alternativen zu >= anzubieten. Da diese



Der Quellcode für den Compiler und die Standard Library wurden von Apple als Open Source freigegeben (Bild 2)

Direktive zur Compile-Time ausgeführt und geprüft wird, bedeutet das, dass jeglicher Code, der sich nicht innerhalb des passenden auszuführenden Blocks befindet, einfach in Gänze ignoriert wird. Somit wird auch kein unnötiger Overhead erzeugt, wenn in einem Projekt auf die gezeigte Art und Weise unterschiedlicher Code für unterschiedliche Swift-Versionen implementiert wird.

Langfristig wird dieses neue Feature gerade für die genannten Library-Anbieter wohl ein wahrer Segen werden, doch leider ist es zum jetzigen Zeitpunkt quasi noch nutzlos. Das liegt daran, dass der Vorgänger von Swift 2.2 – namentlich bekannt als Swift 2.1 – nichts mit dieser Compiler-Direktive anfangen kann; es versteht sie schlicht und einfach nicht, was auch kein Wunder ist: Schließlich ist dieses Feature in dieser Version und allen Vorgängern davon noch nicht implementiert. Das bedeutet aber, dass es erst seine volle Stärke ausspielen wird, wenn die nächste Version von Swift – sehr wahrscheinlich dann Swift 3 – offiziell veröffentlicht ist. Dann kann eine konkrete und sinnvolle Swift-Versionsunterscheidung getroffen werden.

Neue Selector-Direktive

Ein weiteres neues Feature von Swift 2.2 löst zugleich eines der größten Probleme der Vorgängerversionen. Dabei geht es um die Arbeit mit Selectors, wie sie typischerweise zum Angeben von Methodennamen verwendet werden. In Objective-C wurde für diese Zwecke ein entsprechender Befehl verwendet, um einen Methodennamen als Selector zu übergeben. Dieser Befehl hatte den Vorteil, dass er vom Compiler geprüft werden konnte und dadurch festgestellt wurde, ob es die entsprechend angegebene Methode tatsächlich gibt.

In Swift fehlte bisher eine derartige Funktion. Stattdessen sollten Selectors als einfache Strings definiert werden. Das folgende Listing zeigt ein typisches Beispiel für eine derartige Deklaration eines Selectors in bisherigen Swift-Versionen:

```
override func viewDidLoad() {
  super.viewDidLoad()
```

▶

Apples Programmiersprachen

Apple hat eine lange Tradition auf dem Gebiet der Programmiersprachenentwicklung.

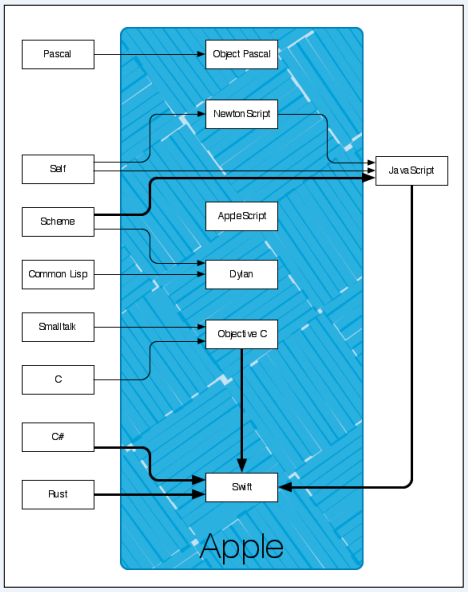
AppleScript ist vielleicht das naheliegendste Beispiel für eine von Apple entworfene Skriptsprache. Die Sprache entstand Ende der 1980er Jahre im Rahmen von HyperCard, einem der ersten Hyper-Text-Systeme.

Ebenfalls Ende der 1980er Jahre kaufte Apple die Common Lisp Implementierung Coral Common Lisp und entwickelte die hochdynamische und gleichzeitig zu effizientem Maschinencode kompilierende Sprache als Macintosh Common Lisp (MCL) weiter.

Da die Entwicklung der Sprache für das Newton-Projekt nicht schnell genug vorankam, wurde das Betriebssystem des Handhelds stattdessen mit einer Mischung aus C und einer wiederum neuen Skriptsprache namens NewtonScript entwickelt.

Auch das klassische Mac OS wurde hauptsächlich in einer ebenfalls bei Apple beheimateten Programmiersprache entwickelt: Object Pascal. Mitte der 1990 Jahre wechselte Apple – wohl auch auf den Druck des Mainstream-Entwicklermarkts hin – auf C++.

Nachdem Steve Jobs 1985 aus seinem eigenen Unternehmen verdrängt wurde, schuf er mit seiner neuen Firma NeXT einen neuen Rechner mit einer von Grund auf neu konzipierten objektorientierten Plattform. Als Programmiersprache setzte NeXT auf das bis dahin noch eher unbekannte Objective-C. Mit Jobs Rückkehr zu Apple wurde aus der Plattform NextStep die Grundlage für Mac OS X, und Objective-C wurde die neue Standard-Entwicklungssprache der neuen Generation von Macs und damit später auch der iOS-Plattform.



Apple hat eine lange Tradition in der Entwicklung von Programmiersprachen

```
  navigationItem.rightBarButtonItem =
  UIBarButtonItem(barButtonSystemItem: .Add, target:
  self, action: "doSomthing")
}
func doSomething() {
  print("Do something...")
}
```

Aufmerksamen Lesern wird wohl der Fehler aufgefallen sein, der sich eingeschlichen hat und der gleichzeitig das größte Problem und die größte Schwäche der bisherigen Deklaration von Selectors in Swift aufzeigt. Die Methode, die nämlich als Selector für das *UIBarButtonItem* definiert wird, gibt es so nicht. Statt des korrekten Methodennamens *doSomething* lautet der Name des Selectors nun *doSomthing;* das e fehlt an dieser Stelle. In Objective-C wäre das kein Problem, da der genannte Selector-Befehl vom Compiler geprüft worden wäre und dieser festgestellt hätte, dass es die aufzurufende Methode *doSomthing* gar nicht gibt. In Swift aber lässt der Compiler das durchgehen, das böse Erwachen erfolgt erst zur Laufzeit.

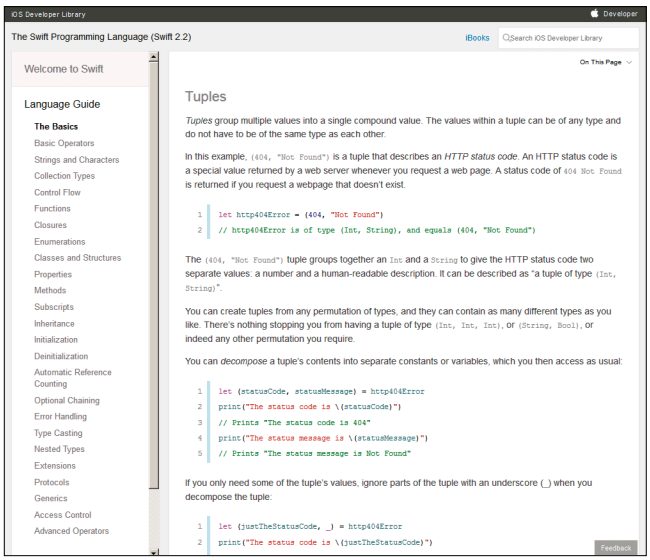
Compiler-Prüfung

Die fehlende Compiler-Prüfung von Selectors war von Beginn an eine der Schwachstellen von Swift im Vergleich zu Objective-C. Man musste immer aufpassen, den richtigen Methodennamen als String für einen Selector zu schreiben. Ein späteres mögliches Umbenennen von Methoden, deren zuvor erstellte Selectors dann ins Leere führten, war da dann die Krönung für dieses Problem, bei dem sich wohl der ein oder andere Entwickler schon genervt die Haare gerauft hat.

Mit Swift 2.2 bessert Apple an dieser Stelle aber nach, und das absolut konsequent: Der im Beispiel gezeigte Code ist in Swift 2.2 deprecated und führt zu einer Compiler-Warnung. Statt des Selector-Strings soll ab Swift 2.2 nämlich die neu eingeführte *#selector*-Direktive für alle Arten von Selectors verwendet werden. Diese ist quasi identisch zum Objective-C-Befehl zum Erstellen von Selectors. Nach der genannten Direktive folgt innerhalb von runden Klammern der Name der Methode, auf die der Selector verweisen soll. Und dank der neuen Direktive kann nun auch der Compiler feststellen, ob die übergebene Methode tatsächlich existiert oder nicht. Das folgende Listing zeigt dazu noch einmal den entsprechend überarbeiteten Code aus dem vorigen Beispiel:

```
override func viewDidLoad() {
  super.viewDidLoad()
  navigationItem.rightBarButtonItem =
  UIBarButtonItem(barButtonSystemItem: .Add, target:
  self, action: #selector(doSomething))
}
func doSomething() {
  print("Do something...")
}
```

Auch wenn diese neue Selector-Direktive auch in Swift 2.2 noch nicht zwingend verwendet werden muss, so ist davon auszugehen, dass Apple spätestens mit Swift 3 das Verwenden von Strings als Selectors verbieten wird. Es ist daher absolut sinnvoll, so bald wie möglich auf *#selector* umzusteigen, natürlich auch nicht zuletzt deswegen, um die beschriebenen Probleme zu umgehen.



Tupel erlauben die Definition mächtiger Typen (Bild 3)

Schlüsselwörter für External Parameter Names

External Parameter Names sind der Teil einer Methode, der aufgeführt werden muss, wenn diese Methode aufgerufen wird. Dabei war es bisher nicht ohne Weiteres möglich, von Swift verwendete Schlüsselwörter als External Parameter Names zu verwenden, was im Allgemeinen auch nicht weiter problematisch ist, schließlich sind derartige Schlüsselwörter wie *class* oder *func* eben für die Programmiersprache reserviert. Dennoch ist es in manchen Fällen sinnvoll, reservierte Schlüsselwörter von Swift als External Parameter Name zu verwenden. Wollte man das bisher tun, musste man dazu das entsprechende Keyword in Anführungszeichen setzen.

Mit Swift 2.2 hat Apple dieses Verhalten ein wenig optimiert. Nun ist es nämlich möglich, direkt alle Schlüsselwörter als External Parameter Names zu verwenden, ohne diese in Anführungszeichen setzen zu müssen, solange es sich nicht um die Keywords *inout, var* und *let* handelt. Diese drei können zwar auch als External Parameter Names verwendet werden, müssen aber weiterhin explizit in Anführungszeichen gesetzt werden:

```
func visitCity(name: String, in state: String) {
  print("I'm going to visit \(name) in \(state)")
}
visitCity("Aschaffenburg", in: "Bavaria")
```

Tupel spielen in Swift eine große Rolle und erlauben es, mächtige Typen zu definieren, die mehrere verschiedene Werte in einer geordneten Struktur zusammenfassen (Bild 3). Mit ihnen ist es beispielsweise einfach und übersichtlich möglich, eine Funktion mit mehreren verschiedenen Rückga-

bewerten zu definieren, indem die entsprechenden Werte in ein dafür passendes Tupel gepackt werden.

So mächtig Tupel auch sind, so kompliziert und aufwendig war es bisher, zwei Tupel miteinander zu vergleichen. Wollte man jedes Element eines Tupels mit dem zugehörigen Element eines anderen Tupels vergleichen, blieb einem nichts anderes übrig, als das selbst Element für Element zu tun und anschließend entsprechend zu reagieren.

Mit Swift 2.2 hat Apple an dieser Stelle nachgebessert und erlaubt nun das bequeme Vergleichen von zwei Tupeln über die bekannten Vergleichsoperatoren == und !=. Swift übernimmt dann die Aufgabe, die einzelnen Elemente der Tupel miteinander zu vergleichen, und liefert dann entweder *true* zurück, wenn alle verglichenen Elemente miteinander identisch sind, oder *false*, wenn nicht. Listing 1 zeigt dazu einmal passende Beispiele.

Beim Vergleich von Tupeln ist aber zu beachten, dass tatsächlich nur die Werte der einzelnen Elemente miteinander verglichen werden, nicht die Namen der zugehörigen Parameter. Listing 2 zeigt dazu ein ergänzendes Beispiel, in dem zwar die Namen der Parameter der miteinander verglichenen Tupel unterschiedlich, die jeweiligen Elemente aber identisch sind. Entsprechend ist der Vergleich der beiden Tupel trotz unterschiedlicher Parameternamen positiv.

So schön diese neue Möglichkeit, Tupel miteinander zu vergleichen, in Swift 2.2 auch ist, so muss man eine kleine Einschränkung dennoch in Kauf nehmen: Die gezeigte Möglichkeit des Tupel-Vergleichs funktioniert nur bei Tupeln bis maximal sechs Elementen. Verfügen Tupel über mehr als sechs Elemente, ist ein Vergleichen auf die gezeigte Art und Weise auch in Swift 2.2 nicht mehr möglich.

Listing 1: Vergleich von Tupeln

```
let thomas = (first: "Thomas", last: "Sillmann")
let clone = (first: "Thomas", last: "Sillmann")
let max = (first: "Max", last: "Mustermann")
if thomas == clone {
  print("thomas and clone are identical!")
} else {
  print("thomas and clone are not the same.")
}
// thomas and clone are identical!
if thomas == max {
  print("thomas and max are identical!")
} else {
  print("thomas and max are not the same.")
}
// thomas and max are not the same.
```

Listing 2: Vergleich von Tupeln

```
let anotherThomas = (first: "Thomas",
last: "Sillmann")
let anotherClone = (firstName: "Thomas",
lastName: "Sillmann")
if anotherThomas == anotherClone {
  print("anotherThomas and anotherClone are
  identical!")
} else {
  print("anotherThomas and anotherClone are not
  the same.")
}
// anotherThomas and anotherClone are identical!
```

Entfernung der Tuple Splat Syntax

Selbst wer schon lange Zeit mit Swift entwickelt, kennt womöglich nicht den Begriff der Tuple Splat Syntax. Dabei handelt es sich um eine Funktionalität von Swift, die es erlaubt, ein Tupel als einzigen Parameter einer Methode zu übergeben, sofern das Tupel genau über dieselben Parameter verfügt wie die aufzurufende Methode:

```
func describePerson(name: String, age: Int) {
  print("\(name) is \(age) years old")
}
let person = ("Thomas Sillmann", age: 27)
describePerson(person)
```

Diese Funktionalität wurde in Swift tatsächlich kaum verwendet, weshalb Apple sich nun auch dazu entschloss, sie vollumfänglich aus Swift zu entfernen. Entsprechend hat die Tuple Splat Syntax ab Swift 2.2 ausgedient und wird dort als deprecated gekennzeichnet.

for-Schleifen

Eine große Änderung hat Apple im Bereich der *for*-Schleifen vorgenommen. Bisher konnten diese ähnlich wie in C deklariert werden, indem man bei der Deklaration einer *for*-Schleife nacheinander eine Initialisierung, eine Bedingung und ein Inkrement deklarierte, die einzelnen Elemente jeweils durch Semikolon getrennt. Doch damit ist ab Swift 2.2 Schluss. Offensichtlich könnte man meinen, Apple störte sich dabei schlicht an den Semikolons, schreiben sie doch zur entsprechenden Bekanntgabe in einem Blog-Beitrag: »One more step towards never typing a semi-colon again«. Doch was heißt das konkret?

Generell werden die bisherigen *for*-Schleifen nun ab Swift 2.2 vollumfänglich durch die ebenfalls bereits bekannten *for in*-Schleifen ergänzt und gehen darin auf. Ein einfaches Beispiel zeigt zunächst eine alte Variante einer typischen *for*-Schleife, gefolgt von der nun stattdessen zwingend zu verwendenden Variante:

```
// Bisher mögliche Variante einer for-Schleife
for var i = 0; i < 10; i++ {
  print(i)
}
// Nun einzig zulässige Variante einer for-Schleife
// ab Swift 2.2
```

```
for i in 0 ..< 10 {
  print(i)
}
```

Generell ist die nun von Swift 2.2 vorausgesetzte Form der *for*-Schleife nichts Neues, derartige Konstrukte ließen sich bereits mit der ersten Version von Swift implementieren. Wie beim Durchlaufen eines Arrays mittels Fast Forward wird hier ein definierter Wertebereich durchlaufen, im gezeigten Beispiel eine Schleife von 0 bis ausschließlich 10. Das *bis ausschließlich* wird durch das ..< im Code definiert, ein *bis einschließlich* würde in Swift mittels ... umgesetzt.

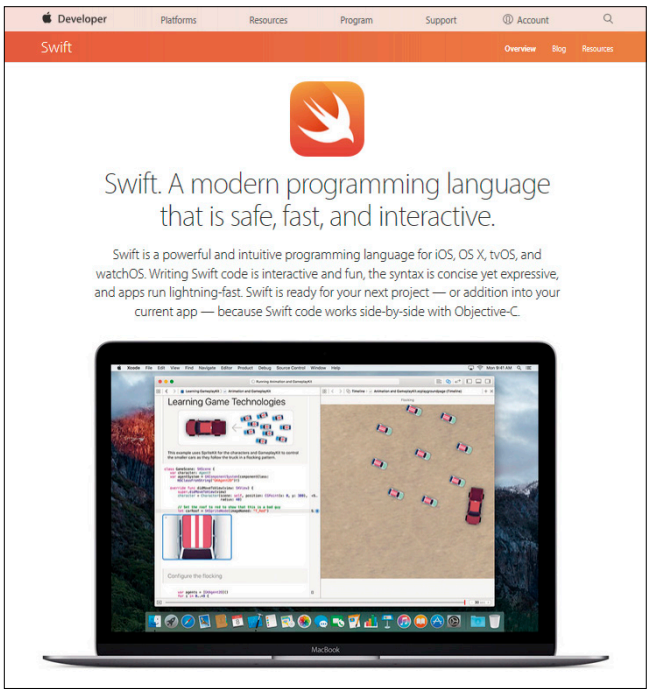
Allerdings lassen sich auf diese Art und Weise nicht alle Arten von *for*-Schleifen realisieren. Was ist beispielsweise, wenn ein Wertebereich nicht hochgezählt, sondern heruntergezählt werden soll, ähnlich einem Countdown? Oder wenn ein Wertebereich nicht in einfachen Schritten – sprich einem Inkrement +1 –, sondern in Zweier- oder Dreier-Schritten durchlaufen werden soll? Beides kann auf die nun einzige zulässige Art und Weise einer *for*-Schleife in Swift 2.2 nicht realisiert werden.

Aus diesem Grund stehen separate Funktionen zur Verfügung, mit denen sich der Wertebereich definieren lässt. Nehmen wir dabei zunächst einmal das genannte Beispiel des Herunterzählens eines Wertebereichs. In Swift 2.2 wird der entsprechende Wertebereich dabei zunächst wie gewohnt definiert, und zwar vom niedrigsten zum höchsten Wert, was dem bekannten Hochzählen entspricht. Anschließend wird auf diesen Wertebereich aber die Funktion *reverse* aufgerufen. Damit dreht Swift den Wertebereich um und zählt vom höchsten Wert zum niedrigsten Wert herunter.

Der zweite Problemfall – das Umsetzen eines Inkrements, das nicht +1 entspricht – ist ein wenig komplizierter umzusetzen. Ähnlich wie bei der Reverse-for-Schleife kommt hierbei eine Funktion von Swift zum Einsatz, die sich *stride(to:by:)* nennt. Der erste Parameter entspricht dem gewünschten Höchstwert des zu durchlaufenden Wertebereichs, während der zweite Parameter die Schrittweite definiert, in der der Wertebereich durchlaufen werden soll. Aufgerufen wird diese Funktion auf den gewünschten Startwert des Wertebereichs.

Listing 3: Kopie eines Parameters statt var-Parameter

```
// Methode mit var-Parameter (bis Swift 2.2)
func greet(var name: String) {
  name = name.uppercaseString
  print("Hello, \(name)!")
}
// Erstellen einer Kopie eines Parameters statt
// var-Parameter (ab Swift 2.2)
func greet(name: String) {
  let uppercaseName = name.uppercaseString
  print("Hello, \(uppercaseName)!")
}
```



Apple drückt bei der Weiterentwicklung von Swift weiter aufs Tempo (Bild 4)

Was womöglich ein wenig kompliziert klingt, soll ein konkretes Beispiel abschließend verdeutlichen. Im folgenden Listing sind wieder zwei *for*-Schleifen aufgeführt. Erstere zeigt das genannte Beispiel in der C-typischen Form, während die zweite Variante die nun als einzige in Swift 2.2 erlaubte Form darstellt:

```
// Bisher mögliche Variante zum Anpassen des Inkrements
for var i = 0; i < 10; i += 2 {
  print(i)
}

// Nun einzig zulässige Variante zum Anpassen des
// Inkrements ab Swift 2.2
for i in 0.stride(to: 10, by: 2) {
  print(i)
}
```

Entwickler, die auch in anderen Programmiersprachen unterwegs sind, mag die folgende Änderung von Swift 2.2 doch sehr überraschen: Apple hat die Prefix- und Postfix-Operatoren ++ und -- gestrichen und als deprecated gekennzeichnet; ab Swift 3 werden sie überhaupt nicht mehr funktionieren.

Einer der Hauptgründe für die Streichung dieser Operatoren liegt in den ebenfalls gestrichenen, C-ähnlichen *for*-Schleifen begründet. Dort kamen derartige Operatoren typischerweise mit am häufigsten zum Einsatz. Da diese aber in Swift 2.2 nicht länger erlaubt sind, war es für Apple offenbar fraglich, ob sie überhaupt noch gebraucht werden. Und offensichtlich dachte die Community ähnlich, schließlich wurde diese Änderung – so wie im Übrigen alle anderen auch – nicht von Apple alleine beschlossen.

Im Klartext bedeutet das für uns Entwickler, dass ab Swift 2.2 nur noch Code wie *i += 1* oder *i -= 1* statt *i++* oder *i--* möglich ist. Erfreulicherweise bietet Xcode häufig passende Fix-its an, um entsprechenden Code automatisch auf den neuen Standard in Swift 2.2 zu aktualisieren.

Deprecation von var-Parametern in Methoden

In Swift 2.2 gibt es noch eine weitere Funktion, die Apple inzwischen als deprecated gekennzeichnet hat. Dabei handelt es sich um die Möglichkeit, Parameter einer Methode mit dem Schlüsselwort *var* zu deklarieren. Das führte bis Swift 2.2 dazu, dass der entsprechende Parameter innerhalb der jeweiligen Methode geändert werden konnte, die entsprechende Änderung aber nur innerhalb dieser Methode gültig war.

Apple begründet die Entscheidung für diese Streichung mit einer möglichen Verwirrung der Entwickler, wenn sie derartig implementierten Code sehen. Insbesondere im Zusammenspiel mit dem *inout*-Keyword hält Apple die zusätzliche Verwendung von *var* für riskant und fehleranfällig, da so nicht unbedingt auf den ersten Blick ersichtlich ist, ob die Änderung eines Methodenparameters nur innerhalb der jeweiligen Methode gilt oder ob der eigentliche Quellparameter selbst ebenfalls geändert wird.

Als Alternative zur Verwendung von *var* bei der Deklaration von Methodenparametern empfiehlt Apple nun, einfach eine Kopie des Parameters innerhalb der jeweiligen Methode zu erstellen und diese auf die gewünschte Art und Weise zu ändern. Dieses Vorgehen ist prinzipiell – auch in Swift – nichts Neues, derartiger Code konnte bereits von Beginn an in Swift geschrieben werden.

Listing 3 zeigt noch ein verdeutlichendes Beispiel zu dieser Thematik, in dem es erst eine Beispielmethode mit *var*-Parameter zeigt, in der der übergebene Parameter entsprechend geändert und anschließend ausgegeben wird. Die zweite Variante entspricht dem Swift-2.2-Standard, indem stattdessen eine Kopie des Parameters innerhalb der Methode erstellt, entsprechend angepasst und dann ausgegeben wird.

Umbenennung der Debug Identifier

Swift bringt seit jeher einige einzigartige Identifier mit, die das Debuggen des eigenen Codes erleichtern sollen. Dazu gehören beispielsweise *__FILE__* oder *__LINE__*. Diese können beispielsweise mittels String-Interpolation ausgegeben werden, um nachvollziehen zu können, welche Datei den entsprechenden Code in welcher Zeile ausführt.

Um eine Einheitlichkeit bei der Verwendung von Compiler-Befehlen zu erreichen, wurden diese Debug Identifier nun durch neue Varianten ersetzt. Sie tragen denselben Namen, werden aber nun mittels vorangestelltem Raute-Zeichen deklariert, also beispielsweise *#file* statt *__FILE__* und *#line* statt *__LINE__*. Das folgende Listing stellt die beiden gegenüber und zeigt die geänderte Implementierung:

```
func myMethod() {
  // Debug Identifier bis Swift 2.2
  print("This is on line \(__LINE__) of
  \(__FUNCTION__)")
  // Debug Identifier ab Swift 2.2
  print("This is on line \(#line) of \(#function)")
}
```

Auch für die Debug Identifier bietet Xcode automatisch passende Fix-its an, um bisherigen Code einfach und schnell auf den neuen Swift-2.2-Standard umzustellen. Ab Swift 3 sind die neuen Debug Identifier Pflicht und die alten Varianten werden nicht mehr funktionieren.

Links zum Thema

- Offizielle Swift-Website
  https://swift.org
- Vorstellung der Features von Swift 2.2
  https://swift.org/blog/swift-2-2-new-features
- Apple Entwickler-Portal
  http://developer.apple.com
- Swift-Übersichtsseite bei Apple
  https://developer.apple.com/swift
- Swift Blog
  https://developer.apple.com/swift/blog

Fazit

Apple gibt weiterhin Vollgas, was die Weiterentwicklung der neuen Programmiersprache Swift betrifft, und das inzwischen auch mit einer großen Open Source-Community im Rücken (Bild 4). Dabei ist das finale Release von Swift 2.2 dahingehend sehr besonders, als es das erste Release ist, dessen Änderungen nicht eigenmächtig von Apple alleine, sondern von der gesamten Community beschlossen wurden.

Was dabei auffällt, ist eine strikte und durchgehende Vereinfachung der Sprache. Unnötige Funktionen werden nicht länger behalten, sondern konsequent entfernt. Befehle, die sich auf Debug-Ebene auswirken, werden nun durchgängig mittels #-Zeichen eingeleitet.

Zudem verliert Swift mehr und mehr von seinem aus C übernommenen Overhead, der nicht so recht in die Sprache passen will. Alles in allem entwickelt sich Swift definitiv in die richtige Richtung, und wir dürfen sehr gespannt auf die nächsten Verbesserungen, Optimierungen und Erweiterungen der Sprache sein. ■

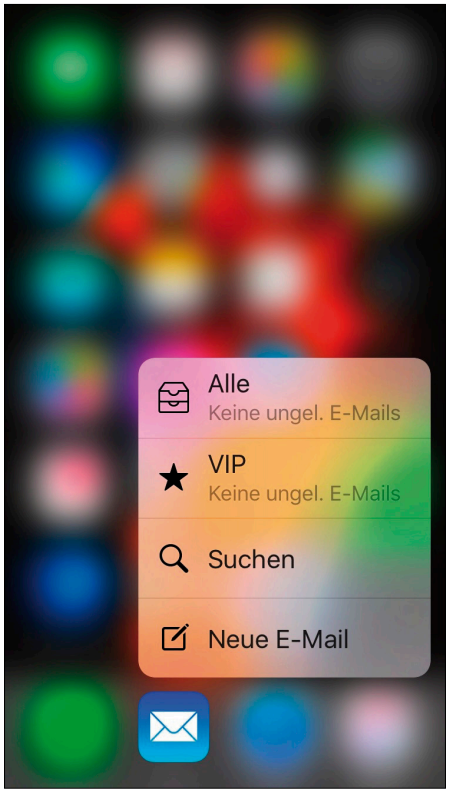
Thomas Sillmann
ist iOS-App-Entwickler, Trainer und Autor. Freiberuflich tätig programmiert er für den App Store eigene Apps sowie Apps in Form von Kundenaufträgen. Er ist Autor eines erfolgreichen Fachbuchs und mehrerer Artikel in Fachzeitschriften.
www.thomassillmann.de

APPLES 3D TOUCH IM IPHONE

Under Pressure

Wie man Apples 3D Touch innerhalb einer App nutzen kann.

Das iPhone 6s ist nicht nur eine technisch aktualisierte Version des iPhone 6, sondern bringt auch einige neue Funktionen mit. Unter dem Namen 3D Touch hat Apple in diese Version seines Smartphones einen drucksensitiven Bildschirm integriert. Das bedeutet, dass das Display beziehungsweise eine im Display zusätzlich integrierte Schicht misst, wie stark ein Anwender auf das Display drückt. Darüber hinaus erhält der Anwender, sofern er fester auf das Display drückt, nicht nur eine visuelle, sondern auch eine haptische Rückmeldung.



Mail-App: Die Shortcuts der Mail-App von iOS (Bild 1)

Taptic Engine

Möglich wird dies durch Apples Taptic Engine, die sich nicht nur im iPhone, sondern auch in der Apple Watch oder in den Touchpads von Macintoshs der neuesten Generation befindet. Die Taptic Engine sorgt dafür, dass eine fühlbare Rückmeldung an den Anwender übertragen wird, sobald das Display des iPhone 6s oder eines der neuen Touchpads gedrückt wird. In Abhängigkeit von der Stärke des Drucks, mit der das Display betätigt wird, können unterschiedliche Rückmeldungen gegeben werden. Ermöglicht wird dieses Verhalten durch einen Vibrationsmotor, der unter dem Display/Touchpad angebracht ist. Wie wirkt sich diese neue Technik nun unmittelbar aus?

Damit iOS beziehungsweise eine App erkennt, ob stark oder schwach gedrückt wird, wurde das Betriebssystem angepasst. Berührt man das Display eines iPhones an einer Stelle nur leicht, so wird eine App, zum Beispiel Mail, so wie bisher auch einfach gestartet. Drückt man hingegen etwas fester, so wird nun ein Shortcut-Menü angezeigt und gleichzeitig ist ein haptisches Feedback spürbar (Bild 1).

Zu sehen ist, dass nun einige Funktionen der Mail-App verfügbar sind, ohne dass die App gestartet wurde, beispielsweise kann so direkt die Funktion *Neue E-Mail* aktiviert werden. Wird der entsprechende Shortcut betätigt, so wird die App im entsprechenden Modus gestartet. Eine weitere Funktion ist das sogenannte Peek & Pop, das auch in der Mail-App zum Einsatz kommt. Hierbei erzeugt ein leichter Druck auf eine Mail in der Liste ein Vorschaufenster, in dem dann der Inhalt angezeigt wird. Drückt man anschließend fester auf das Vorschaufenster, so wird die Mail komplett geöffnet. Die Peek & Pop-Funktion wird dabei von einem haptischen Feedback unterstützt. Man bekommt das Gefühl, dass sich die Mail durch Durchdrücken öffnen lässt. Das ist aber nicht alles, was sich mit 3D Touch umsetzen lässt.

3D Touch im Code

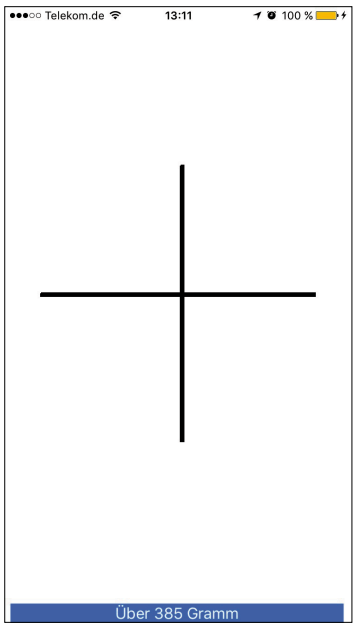
Was ist möglich? Vieles – von interessenten Funktionen wie zum Beispiel den Shortcuts bis hin zu unsinnigen Apps wie einer Waage. Apple hat eine solche Funktion als App bereits abgelehnt. Hierbei sollte man auch berücksichtigen, dass 3D Touch nicht als Ausschlussfunktion für iOS-Geräte fungieren darf. Das heißt, eine Funktion muss sich auch anders realisieren lassen oder aber es darf nicht stören, wenn sie fehlt.

Im Fall der Shortcuts ist es ja kein Problem, wenn diese Funktion zum Beispiel auf einem iPad nicht zur Verfügung steht. Die Mail-App lässt sich trotzdem einwandfrei verwenden. Hier sollte man auch im Hinterkopf behalten, dass die zuletzt vorgestellten iOS-Geräte (iPad Pro oder auch das iPhone SE) nicht über 3D Touch verfügen.

Zurzeit unterstützen unter iOS/watchOS nur das iPhone 6s, das iPhone 6s Plus und die Apple Watch diese Funktion. Wahrscheinlich ist aber, dass zukünftig erscheinende Topmodelle (zum Beispiel das iPhone 7) diese Funktion ebenfalls bieten werden. Ein wichtiger Punkt im Code ist deshalb die Stelle, an der geprüft wird, ob 3D Touch auf dem Gerät verfügbar ist. Der folgende Code kann für diese Abfrage verwendet werden:

```
if traitCollection.
  forceTouchCapability ==
  UIForceTouchCapability.Available {
}
```

Die Klasse *UITraitCollection* beziehungsweise ein Objekt davon kann in einer App verwendet werden, um Informationen über die verwendete Hardware abzufragen. Seit Xcode 7 beziehungsweise Swift 2.0 gibt es in dieser Klasse die Eigen-



Die Waage-App im Einsatz (Bild 2)

schaft *ForceTouchCapability*. Nachdem geklärt wurde, ob das aktuelle iOS-Gerät 3D Touch unterstützt, muss man noch wissen, wo und wann Abfragen zum ausgeführten Druck auf das Display eines Anwenders durchgeführt werden können. Das Betriebssystem bietet hierfür drei Ereignisse an: *touchesBegan*, *touchesMove* und *touchesEnded*.

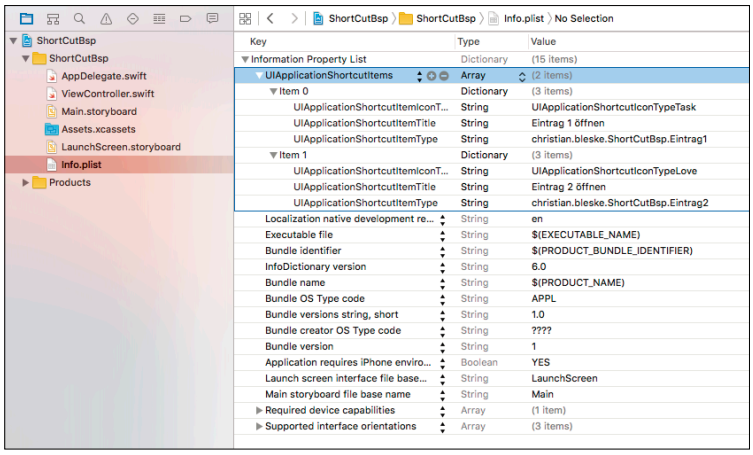
Innerhalb dieser Ereignisse ist es möglich, auf eine Auflistung vom Typ *UITouch* zuzugreifen; diese wird als Parameter übergeben. Diese Auflistung enthält Objekte, die Informationen zur Position, der Größe und zum ausgeübten Druck einer Geste umfassen, und kann im Code natürlich abgefragt werden:

```
print("Der ausgeführte Druck: \(touch.force),
maximaler Druck: \(touch.maximumPossibleForce)")
```

Die Eigenschaft *force* enthält den Wert für den auf das Display ausgeübten Druck. In *maximumPossibleForce* hingegen ist der Wert für den maximal messbaren Druck enthalten. Mit diesem Vorwissen kann man dann loslegen.

Eine Waage mit 3D Touch

Obwohl Apps mit dieser Funktionalität von Apple bereits für den App Store abgelehnt worden sind, verdeutlichen sie die Funktionsweise der neuen Technik doch sehr schön. Als Ausgangspunkt für eine solche App wird eine Single-View-Application-Vorlage gewählt.



UIApplicationShortcutItems in der *Info.plist*-Datei (Bild 3)

Listing 1: Code der Waage-App

```
import UIKit
class ViewController: UIViewController {
  @IBOutlet var uiLabel: UILabel!
  override func viewDidLoad() {
    super.viewDidLoad()
  }
  override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
  }

  override func touchesMoved(touches: Set<UITouch>,
  withEvent event: UIEvent?) {
  if let touch = touches.first {
    if traitCollection.forceTouchCapability ==
    UIForceTouchCapability.Available {
      if touch.force >= touch.maximumPossibleForce {
          uiLabel.text = "Über 385 Gramm"
        } else {
          let result =
          touch.force/touch.maximumPossibleForce
          let calculated = result * 385
          let rounded = Int(calculated)
          uiLabel.text = "\(rounded) Gramm"
        }
      }
    }
  }
  override func touchesEnded(touches: Set<UITouch>,
  withEvent event: UIEvent?) {
    uiLabel.text = "0 Gramm"
  }
}
```

Dem Projekt wird dann ein Bild hinzugefügt, das als Hintergrundbild in einer ImageView verwendet wird. Außerdem wird in der View am unteren Rand noch ein Label-Control eingefügt, das dazu verwendet wird, den auf das Display ausgeübten Druck anzuzeigen (Bild 2).

Nach der Anlage des Projekts wird zunächst eine ImageView und dann ein Outlet für das Label-Control erzeugt. Anschließend kann es ans Programmieren der beiden für die Funktion benötigten Ereignismethoden gehen (Listing 1). Der größte Teil des Codes befindet sich in der Ereignismethode *touchesMove*. Es soll so lange der Druck auf das Display gemessen werden, bis dieser nicht mehr vorhanden ist.

Die beiden anderen Ereignismethoden *touchesBegan* und *touchesEnded* eignen sich für diese Anforderung eher nicht. Innerhalb der Methode *touchesMove* wird im ersten Schritt geprüft, ob auf dem Display eine Geste ausgeführt wurde. Anschließend folgt eine Prüfung, ob das iOS-Gerät 3D Touch unterstützt. Wenn das der Fall ist, geht es weiter. Mit ▶

der folgenden Fallunterscheidung wird geprüft, ob der ausgeführte Druck den maximalen Wert überschreitet. Bewegt sich der Druck im messbaren Rahmen, so geht es im *else*-Zweig weiter. Der auf das Display ausgeübte Druck wird in der folgenden Zeile aus der Eigenschaft *force* ausgelesen und durch den maximal messbaren Wert geteilt. Das Ergebnis wird mit dem Wert 385 multipliziert und zuletzt noch einmal gerundet. So wird der ursprünglich gemessene Wert in einen Gramm-Wert umgewandelt.

Zuletzt wird der ermittelte Gramm-Wert in dem UILabel ausgegeben. Nimmt man das Gewicht wieder vom Bildschirm, so wird in der Ereignismethode der Inhalt des UILabels auf den Wert *0* gesetzt. Mit wenig Aufwand hat man so eine Waage – natürlich nur für kleine Gewichte.

Shortcut definieren

Zu Beginn wurde bereits die Shortcut-Funkton von 3D Touch vorgestellt. Diese Shortcuts oder QuickActions sind eine interessante Möglichkeit, die Bedienung von iOS-Apps weiter zu verfeinern.

Im folgenden Beispiel soll eine App um die Möglichkeit erweitert werden, auf leichten Druck hin zwei Shortcuts zum optionalen Start der Anwendung anzuzeigen.

Als Testballon wird wieder eine Single-View-Application-Vorlage gewählt. Im ersten Schritt müssen die Shortcuts, die nach leichtem Druck auf das Icon angezeigt werden sollen, konfiguriert werden. Dies geschieht durch Erweiterung der Datei *Info.plist*. Pro Shortcut muss im Abschnitt *UIApplicationShortcutItem* ein Eintrag (*Item*) angelegt werden. *UIApplicationShortcutItem* ist ein Array. Die einzelnen Items sind Dictionaries, die als Elemente Strings verwenden (Bild 3).

Ein Item (Shortcut) wird mit Hilfe von mehreren Einträgen definiert. In unserem Beispiel werden *UIApplicationShortcutItemIconType*, *UIApplicationShortcutItemTitle* und *UIApplicationShortcutItemType* verwendet (Listing 2). Im Attribut *UIApplicationShortcutItemType* wird die Bezeichnung des

Listing 2: Shortcut-Konfiguration als XML

```
<key>UIApplicationShortcutItems</key>
<array>
  <dict>
    <key>UIApplicationShortcutItemIconType</key>
    <string>UIApplicationShortcutIconTypeTask
    </string>
    <key>UIApplicationShortcutItemTitle</key>
    <string>Eintrag 1 öffnen</string>
    <key>UIApplicationShortcutItemType</key>
    <string>christian.bleske.ShortCutBsp.Eintrag1
    </string>
  </dict>
  <dict>
    <key>UIApplicationShortcutItemIconType</key>
    <string>UIApplicationShortcutIconTypeLove
    </string>
    <key>UIApplicationShortcutItemTitle</key>
    <string>Eintrag 2 öffnen</string>
    <key>UIApplicationShortcutItemType</key>
    <string>christian.bleske.ShortCutBsp.Eintrag2
    </string>
  </dict>
</array>
```

Listing 3: Methode didFinishLaunchingWithOptions

```
class AppDelegate: UIResponder,
UIApplicationDelegate {
  var window: UIWindow?
  var shortcutItem: UIApplicationShortcutItem?
  func application(application: UIApplication,
  didFinishLaunchingWithOptions launchOptions:
  [NSObject: AnyObject]?) -> Bool {
    print("Funktion didFinishLaunchingWithOptions
    wird ausgeführt...")
    var performShortcutDelegate = true
    if let shortcutItem = launchOptions?
    [UIApplicationLaunchOptionsShortcutItemKey] as?
    UIApplicationShortcutItem {
      print("App via shortcut gestartet")
      self.shortcutItem = shortcutItem
      performShortcutDelegate = false
    }
    return performShortcutDelegate
  }
}
```

Listing 4: Verarbeitung des Shortcuts

```
func handleShortcut(
shortcutItem:UIApplicationShortcutItem ) -> Bool {
  print("Shortcut wird ausgeführt...")
  var result = false
  if( shortcutItem.type ==
  "christian.bleske.ShortCutBsp.Eintrag1" )
  {
    print("- Es wurde Shortcut \(shortcutItem.type)
    ausgeführt.")
    result = true
  }
  if( shortcutItem.type ==
  "christian.bleske.ShortCutBsp.Eintrag2")
  {
    print("- Es wurde Shortcut \(shortcutItem.type)
    ausgeführt.")
    result = true
  }
  return result
}
```

```
Funktion didFinishLaunchingWithOptions wird ausgeführt...
Color should set...
Funktion performActionForShortcutItem wird ausgeführt...
Shortcut wird ausgeführt...
- Es wurde Shortcut christian.bleske.ShortCutBsp.Eintrag1 ausgeführt.
Funktion performActionForShortcutItem wird ausgeführt...
Shortcut wird ausgeführt...
- Es wurde Shortcut christian.bleske.ShortCutBsp.Eintrag2 ausgeführt.
```

Verabeitung des Codes nach Aufruf eines Shortcuts (Bild 4)

Shortcuts hinterlegt, mit der sich der Shortcut im Code ansprechen lässt.

Es gibt noch weitere Elemente, die zur Konfiguration eines Shortcuts verwendet werden können. So können pro Shortcut beispielsweise noch ein *Subtitle* sowie eine *UserInfo* erfasst werden. Im Kasten »Links zum Thema« finden Sie den passenden Link zur Apple-Dokumentation der Shortcuts.

Startet man die App, so werden die Shortcuts bereits angezeigt, nachdem diese angelegt wurden. Betätigt man den Shortcut, passiert natürlich noch nichts.

Shortcuts: Die im Beispiel definierten Shortcuts (Bild 5)

Shortcut im Code

Das komplette Handling der Shortcuts findet in der Klasse *AppDelegate* eines Projekts statt. In mehreren Methoden der Klasse muss zur Bearbeitung eines Aufrufs der App über einen Shortcut Code hinterlegt werden. Los geht es in der Ereignismethode *didFinishLaunchingWithOptions* (Listing 3).

In der Methode wird zuerst ein Marker *(performShortcutDelegate)* gesetzt. Anschließend wird in der folgenden Zeile geprüft, ob die App über einen Shortcut gestartet worden ist. Wenn das der Fall ist, wird eine entsprechende Meldung ausgegeben und der verwendete Shortcut wird der Variablen *shortcutItem* übergeben.

Dem Marker wird zuletzt ein *false*-Wert zugewiesen. Dieser Wert wird anschließend als Rückgabewert der Methode verwendet. Im nächsten Schritt wird die Methode *performActionShortcutItem* ausgeführt:

```
func application(application: UIApplication,
performActionForShortcutItem shortcutItem:
UIApplicationShortcutItem, completionHandler: (Bool) ->
Void) {
    print("Funktion
    performActionForShortcutItem wird
    ausgeführt...")
    completionHandler( handleShortcut
    (shortcutItem) )
}
```

Innerhalb der Methode *performActionShortcutItem* wird lediglich die Verarbeitung an die Methode *handleShortcut* weitergereicht. Als Parameter wird der betätigte Shortcut übergeben.

Die letzten Schritte der Verarbeitung finden innerhalb der Methode *handleShortcut* statt (Listing 4). Nach Ausgabe eines Hinweises wird geprüft, welcher Shortcut vom Anwender betätigt wurde. Im Beispiel ist dies entweder *Eintrag 1* oder *Eintrag 2*. Je nach Auswahl wird der entsprechende Code innerhalb der Fallunterscheidung ausgeführt. In Bild 4 ist der Ablauf der Verarbeitung im Output-Fenster zu sehen.

Bild 5 zeigt die beiden im Beispiel definierten Shortcuts. Auffällig ist, dass neben dem angegebenen Namen noch jeweils ein Icon angezeigt wird. Woher kommt das? Sehen Sie sich hierzu noch einmal Listing 2 an. Dort finden Sie den Eintrag *UIApplicationShortcutIconTypeTask* – mit ihm wird das Icon definiert. Am Ende der Bezeichnung findet sich der Name des jeweils zu verwendenden Icons. Im Beispiel sind dies *Task* und *Love*.

Eine vollständige Auflistung aller systemseitig verfügbaren Icons finden Sie in der Apple-Dokumentation. Aber auch eigene Bilder können verwendet werden. Hierzu müssen diese im Abschnitt *Assets.xcassets* des Projekts hinterlegt und dann in der *Info.plist*-Datei namentlich benannt werden.

Fazit

Durch 3D Touch wird iOS-Apps eine weitere Option der Interaktion hinzugefügt. Mit dem passenden Wissen lassen sich Apps sehr einfach an die neue Technologie anpassen. ■

Links zum Thema

- Framework Reference
  https://developer.apple.com
- UIApplicationShortcutItem
  https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplicationShortcutItem_class
- UIApplicationShortcutIcon-List
  https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplicationShortcutIcon_Class/#//apple_ref/c/tdef/UIApplicationShortcutIconType

Christian Bleske
ist Autor, Trainer und Entwickler mit dem Schwerpunkt Client / Server und mobile Technologien. Erreichbar ist er unter
cb.2000@hotmail.de

SPIELEPROGRAMMIERUNG UNTER OPENGL (TEIL 3)

Im Namen des Modells

Wirklich interessant wird ein Game erst dann, wenn es auch eine Handlung aufweist.

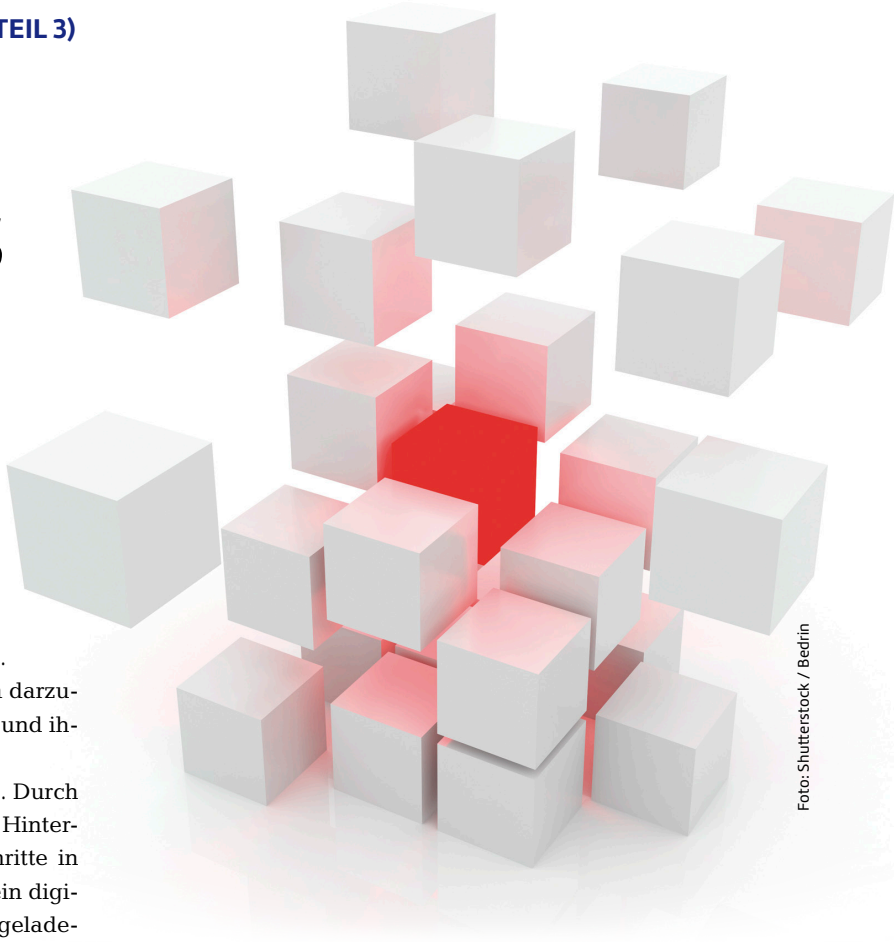


Foto: Shutterstock / Bedrin

Die bisherige Architektur unseres Beispielprogramms ist für den Ausbau zu einem Spiel denkbar ungeeignet, da wir alles direkt rendern. In Engines ist es üblich, die einzelnen am Bildschirm darzustellenden Elemente in Modellklassen zu verpacken und ihre Verwaltung auf diese Weise zu vereinfachen.

Dieser Artikel behebt diesen eklatanten Missstand. Durch die Implementierung von Kollisionserkennung und Hintergrundmusik unternehmen wir wichtige weitere Schritte in Richtung eines vollwertigen Spiels – am Ende steht ein digitaler Kugelkasten, in dem eine aus einer OBJ-Datei geladene Affenfigur von Quadern bedrängt wird.

Funktionsumfang einer Modellklasse

In der Literatur finden sich eine Vielzahl verschiedener Definitionen für den Funktionsumfang einer Modellklasse. In der praktischen Erfahrung des Autors hat es sich empfohlen, die von Microsoft in XNA vorgegebene Struktur zu übernehmen.

Listing 1: Konstruktor

```
public class NMGModelClass {
  public FloatBuffer myPositions;
  ...
  private float[] myPosData;
  ...
  public NMGModelClass(float[] _pos, float[] _col,
  float[] _norm, float[] _tex) {
    myPosData=_pos;
    myPositions = ByteBuffer.allocateDirect
    (_pos.length * 4).order
    (ByteOrder.nativeOrder()).asFloatBuffer();
    myPositions.put(_pos).position(0);
    myColors = ByteBuffer.allocateDirect
    (_col.length * 4).order
    (ByteOrder.nativeOrder()).asFloatBuffer();
    myColors.put(_col).position(0);
    ...
```

Dort ist ein Modell beziehungsweise ein Engine-Element als ein Element deklariert, das neben den Vertexinformationen auch Informationen über die grafische Darstellung und die Position am Bildschirm enthält. Zudem werden relativ häufig Informationen über die Bewegungsrichtung von einer Ableitung eingepflegt. Öffnen Sie als erste Aufgabe die aus dem vorangegangenen Heft bekannte Beispielapplikation und ergänzen Sie sie um eine Klasse namens *NMGModelClass*.

Komplette Abstraktion

Die Aufgabe der neuen Klasse besteht darin, eine komplette Abstraktion beziehungsweise Enkapsulierung aller für die Darstellung eines Modells notwendigen Informationen vorzunehmen. Eine grundlegende Version der Klasse würde folgendermaßen aussehen:

```
public class NMGModelClass {
  float[] myPosData;
  float[] myColorData;
  float[] myNormalData;
  float[] myTexData;
  private float xpos, ypos, zpos;
  float dx, dy, dz;
  public void draw(){}
  public void update(){}
}
```

Neben den diversen Datenfeldern mit Vektorinformationen finden sich hier auch zwei Methoden namens *draw()* und *up-*

date(). Erstere Methode ist für das Zeichnen des Modells, die zweite für seine Positionierung und Bewegung in der Spielwelt zuständig.

Positionieren und Mitdenken

Die Spieleentwicklung gilt unter anderem deshalb als so kompliziert, weil sich fehlende Effizienz an einer ungünstigen Position massiv auf die Gesamtperformance auswirkt. Ein gutes Beispiel dafür wäre eine Modellklasse, die eine Millisekunde Rechenleistung verschwendet. Enthält das Spiel zehn Modelle und sollte mit sechzig Frames dargestellt werden, so verlieren wir 600 Millisekunden.

Eine einfache und zugleich wichtige Methode zur Reduktion des Verbrauchs von Rechenleistung ist das Einsparen unnötiger mathematischer Operationen. Ein gutes Beispiel dafür ist die permanente Regeneration der Modell- beziehungsweise Weltmatrix: Sie muss nur dann erfolgen, wenn sich die Position des Modells auch wirklich geändert hat.

Um dies zu realisieren, setzen wir die Rotations- und Translationsfelder *private*. Wer die Position und die Drehung des Objekts beeinflussen möchte, muss dies fortan über zwei Akzessorfunktionen tun:

```
public class NMGModelClass {
  private float xpos, ypos, zpos;
  private float rotx, roty;
  ...
  public void setNewRot(float _rx, float _ry) {
    rotx=_rx;
    roty=_ry;
    setNewPos(xpos, ypos, zpos);
  }
```

setNewRot ist hier als eine Generalisierung von *setNewPos* implementiert, die die zweite Methode mit einem idempotenten Parametersatz aufruft:

```
  public void setNewPos(float _x, float _y, float _z)
  {
    xpos=_x;
    ypos =_y;
    zpos=_z;
    Matrix.setIdentityM(myModelMatrix, 0);
    Matrix.translateM
    (myModelMatrix, 0, xpos, ypos, zpos);
    Matrix.rotateM(myModelMatrix, 0, rotx,
    0.0f, 1.0f, 0.0f);
    Matrix.rotateM(myModelMatrix, 0,
    roty, 1.0f, 0.0f, 0.0f);
  }
}
```

Diese Accessorfunktionen können den Inhalt der Modellmatrix automatisch aktualisieren, wenn dies erforderlich ist. Da die Gesamtmatrix aus mehreren Multiplikationen entsteht, ist das direkte Ansprechen der Einzelfelder schwierig. Zur Vereinfachung des Entwicklungsprozesses können wir den Konstruktor wie in Listing 1 aufbauen.

Listing 2: onDrawFrame

```
@Override
public void onDrawFrame(GL10 gl) {
  ...
  int aTexID = GLES20.glGetUniformLocation(myFinal
  ProgramHandle, "u_Texture";
  for(int i=0;i<myModelCollection.length;i++) {
    myModelCollection[i].myTextures.position(0);
    GLES20.glVertexAttribPointer(aTexCoordID, 2,
    GLES20.GL_FLOAT, false, 0, myModelCollection[i].
    myTextures);
    ...
    GLES20.glDrawArrays(GLES20.GL_TRIANGLES, 0,
    myModelCollection[i].countTriangles());
  }
}
```

Es fällt auf, dass wir nur eine Referenz auf die Positionsdaten halten. Für alle weiteren Informationen reichen uns die Floatbuffer aus. Das liegt daran, dass wir im Folgenden nur die Positionsdaten verwenden. Sollte Ihre Applikation auch andere Daten brauchen, müssen Sie den Code entsprechend anpassen.

Damit können wir auch schon zur Realisierung der *update()*-Funktion schreiten, die nun so aussieht:

```
public void update() {
  setNewPos(xpos+dx, ypos+dy, zpos + dz);
}
```

In unserem vergleichsweise einfachen Beispiel reicht ein direkter Aufruf von *setnewpos*. Dies wäre in einem praktischen Spiel insofern ineffizient, als bei jedem Durchlauf der Gameloop eine neue Weltmatrix entsteht – auch dann, wenn das Objekt stillsteht.

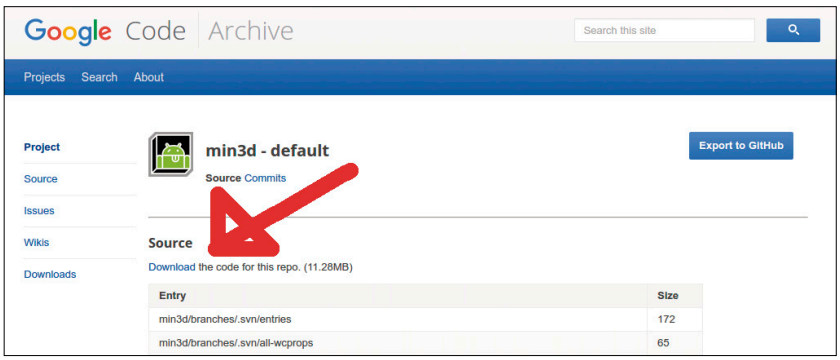
Neben der Nutzung eines *IsStationary*-Flags bietet sich hier auch die Realisierung einer zweistufigen Klassenhierarchie samt verschiedenen *update()*-Implementierungen an.

Mehrere Methoden

OpenGL ES zeichnet sich durch seinen offenen Aufbau aus. Ees gibt für fast jedes Ziel mehrere Methoden zu dessen Erreichung. Im Fall der Erzeugung von Modellklassen ist der Streitpunkt die Verteilung der anfallenden Aufgaben. Der Autor lässt in diesem Beispiel den Gutteil der Konfiguration der Renderingpipeline in *onDrawFrame*. Die Methode muss nun angepasst werden, sodass sie statt eines Modells eine ganze Rotte rendert. Als ersten Akt aktualisieren wir die in den Klassen enthaltenen Informationen zur Physiksituation:

```
public void onDrawFrame(GL10 gl)  {
  rotationStep+=1;
```

```
  GLES20.glClear
  (GLES20.GL_COLOR_BUFFER_BIT |
  GLES20.GL_DEPTH_BUFFER_BIT);
  GLES20.glUseProgram
  (myFinalProgramHandle);
  for(int i=0; i <
  myModelCollection.length;i++) {
    myModelCollection[i].update();
  }
```



Google Projects versteckt den Download-Link gut (Bild 2)

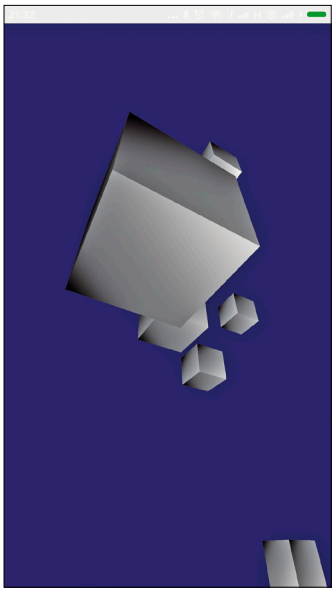
Der eigentliche Zeichenprozess kann währenddessen im Großen und Ganzen vom vorigen Beispiel übernommen werden. Wir durchlaufen alle in der *modelCollection* enthaltenen Modelle und senden die jeweiligen Daten in Richtung der GPU (Listing 2).

OpenGL ES erleichtert uns die Arbeit an dieser Stelle insofern, als die angelieferten Matrices sofort nach dem Aufruf der jeweiligen Methode in Richtung der GPU kopiert werden – es spricht nichts dagegen, die Felder mehrfach weiterzuverwenden. Das gilt auch für die Shaderparameter: Ihre ID bleibt zwischen den einzelnen Aufrufen konstant. Wirklich interessant wird unsere Modellklasse, wenn sie mehrere Objekte gleichzeitig auf den Bildschirm bringt. Fürs Erste reichen uns einige Quader. Hierzu ist in *onSurfaceCreated* eine Codeanpassung notwendig, die sich folgendermaßen präsentiert:

```
@Override
public void onSurfaceCreated(GL10 gl, EGLConfig config)
{
for(int i=0;i<10;i++) {
  myModelCollection[i]=new NMGModelClass
  (NMGCubeConsts.PositionData,
  NMGCubeConsts.ColorData,
  NMGCubeConsts.NormalData,NMGCubeConsts.TextureData);
  myModelCollection[i].dx
  = randFrom(0.01f);
  myModelCollection[i].
  dy = randFrom(0.01f);
  myModelCollection[i].
  dz = randFrom(0.01f);
  myModelCollection[i].
  setNewPos(randFrom(5),
  randFrom(5),
  randFrom(5));
}
```

randFrom() ist eine Conveniencefunktion, die den in Java implementierten Zufallszahlengenerator um eine in einem bestimmten Wertebereich liegende Zufallszahl erleichtert:

```
public float
randFrom(float _high) {
  float oneToZero=2f*(myRandom.nextFloat()-0.5f);
  return oneToZero*_high;
}
```



Die Würfel sind gefallen (Bild 1)

Bei der Entwicklung zufallsgetriebener Applikationen ist es sinnvoll, die *Rand*-Instanz unter Übergabe eines konstanten Seeds zu parametrieren. In diesem Fall ist die zurückgelieferte Zufallszahlenfolge stets identisch, was das Lokalisieren von Fehlern ermöglicht:

```
Random myRandom;
public NMGRenderer(Application _anApp, NMGGLView _aView)
{
  ...
  myRandom.setSeed(232);
}
```

Damit ist die Applikation auch schon zur Ausführung bereit. Bild 1 zeigt die Würfelrotte einige Sekunden nach dem Start des Programms. Zwecks besserem Aussehen wurden die Kameraposition und die Sichtbarkeitsebene angepasst.

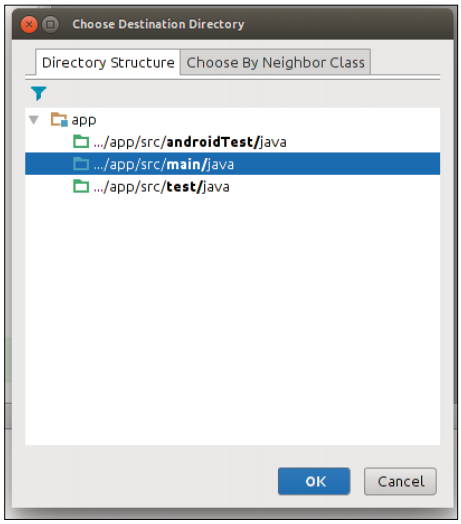
Fortgeschrittene Modellierung

Unsere Würfel entstanden bisher anhand von manuell angelegten Koordinaten-Arrays. Diese für kleinere Gehversuche geeignete Vorgehensweise wird in dem Moment zum Problem, in dem wir komplexere Modelle realisieren wollen.

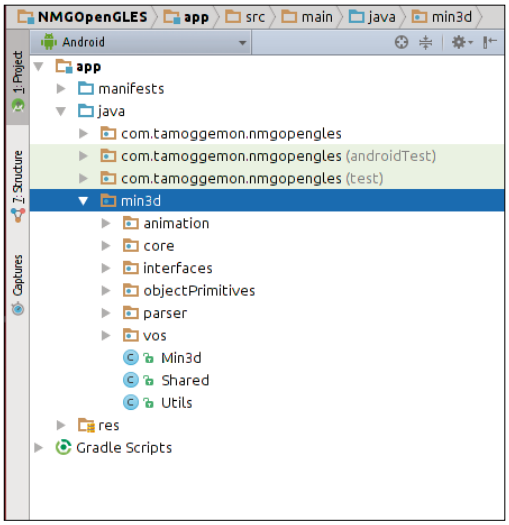
Fertig zugekaufte oder von Grafikern angelieferte Modelle liegen so gut wie immer als OBJ-Dateien vor. Es handelt sich dabei um ein textbasiertes Datenformat, in dem die diversen Koordinaten in einer an CSV erinnernden Art vorgehalten werden.

Auguren raten Quereinsteigern immer wieder zur manuellen Implementierung eines OBJ-Parsers – eine Aufgabe, die nur aus didaktischen Gründen sinnvoll ist. Professionelle Nutzer von OpenGL setzen so gut wie immer auf fertige Modelllader. Min3D ist schon lange verfügbar und gilt als sehr stabil (https://code.google.com/archive/p/min3d/source/default/source). Das Codearchiv ist rund 11 MByte groß (Bild 2).

Die für unser Projekt interessanten Dateien finden sich im Ordner */min3d/trunk/src/min3d/*. Wegen der vergleichsweise komplexen Interdependenzen von Min3D wollen wir die Engine als Ganzes in unser Projekt einbinden. Ein Rechts-



Android Studio: Wohin mit dem Code? (Bild 3)



Min3D wurde ins Projekt integriert (Bild 4)

klick auf den Java-Ordner öffnet ein Kontextmenü, in dem wir die Optionen *New* und *Package* auswählen. Android Studio öffnet daraufhin das in Bild 3 gezeigte Dialogfenster, in dem Sie das Verzeichnis */main/* wählen. Vergeben Sie als Paketname *min3d*, um das neue Containerelement zu erzeugen.

Markieren Sie alle *min3d*-Unterverzeichnisse sowie die drei Codedateien im nächsten Schritt im Dateimanager Ihres Betriebssystems und kopieren Sie sie dann in die Zwischenablage. Klicken Sie das neu erstellte Paket in Android Studio rechts an und wählen Sie die Option *Paste* aus. Nach dem erfolgreichen Einfügen präsentiert sich der Codebaum wie in Bild 4 gezeigt.

Min3Ds Parser weisen eine vergleichsweise komplexe Include-Hierarchie auf. Der für unsere Datei geeignete Parser vom Typ *ObjParser* ist sowohl von *AParser* als auch von IParser abhängig:

```
public class ObjParser extends AParser implements
IParser {
  private final String VERTEX = "v";
  ...
```

Im Rahmen des Ladeprozesses bevölkert die Engine drei globale Felder namens *vertices*, *texCoords* und *normals*. Sie werden in AParser folgendermaßen implementiert:

```
public abstract class AParser implements IParser {
  ...
  protected ArrayList<Number3d> vertices;
  protected ArrayList<Uv> texCoords;
  protected ArrayList<Number3d> normals;
```

Min3D unterstützt keine Materialfarben. Wer derartige Modelle parsen möchte, müsste die dazu notwendige Logik in der *parse()*-Methode von Hand implementieren.

Da unsere Modellklasse mit klassischen Float-Reihen arbeitet, müssen wir die Werte beim Entgegennehmen der Daten umformatieren. Dazu markieren Sie die Typen *Number3d* und *Uv* und wechseln durch das per Rechtsklick einblendbare Kontextmenü zur Deklaration des jeweiligen Elements. *Number3d* sieht dann so aus:

```
public class Number3d {
  public float x;
  public float y;
  public float z;
  ...
```

Die für die Texturkoordinaten zuständige Klasse hört ihrerseits auf den Namen *Uv* und weist eine ähnliche Deklaration auf:

```
public class Uv {
  public float u;
  public float v;
```

Mit diesem Wissen können wir uns der Realisierung eines Parsers zuwenden. Zum Zugriff auf die drei Felder müssen wir in *ObjParser* Akzessoren anlegen, deren Struktur folgendermaßen aussieht:

```
public class ObjParser extends AParser implements
IParser {
  public ArrayList<Number3d> getVertices(){return
  vertices;}
  public ArrayList<Uv> getTexCoords(){return texCoords;}
  public ArrayList<Number3d> getNormals()
  {return normals;}
```

An dieser Stelle wollen wir ein weiteres Ärgernis beheben: Min3D beschafft Ressourcen unter Nutzung der *getIdentifyer*-Methode. Google rät von ihr mittlerweile offiziell ab. Die Entwicklerforen sind voller Beschwerden über die gelinde gesagt eigenwillige Syntax.

Um uns weitergehende Eingriffe in den Code zu ersparen, beschränken wir uns auf das Anpassen der *Parse*-Methode. Entfernen Sie das *override*-Attribut, um die Methode aus der Klassenhierarchie zu lösen. Passen Sie Ihren Header sodann folgendermaßen an:

```
public void parse(int _resID) {
  long startTime = Calendar.getInstance().getTimeI
  InputStream fileIn = resources.openRawResource(_
  BufferedReader buffer = new BufferedReader(
    new InputStreamReader(fileIn));
  String line;
```

Nutzer unserer geänderten Variante von *ObjParser* rufen den Konstruktor des Objekts mit einem beliebigen String auf. Er ist für den hier realisierten Workflow nicht von Relevanz. ▶

Alternativ dazu lässt sich der Parametertyp im Konstruktor anpassen. Die genaue Vorgehensweise ist wie so oft Geschmackssache.

Instanzierung der Objekte

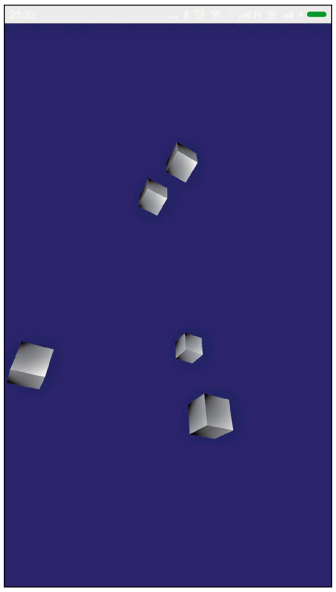
Den eigentlichen Parser legen wir in der Modellklasse ab, um die Instanzierung der Objekte zu vereinheitlichen.

Zum Zugriff auf die Ressourcen sind einige Parameter erforderlich, die die Signatur der zu erzeugenden Funktion bestimmen:

```
public NMGModelClass(Application myApplication,
int resourceID)
{
  ObjParser myParser=new ObjParser
  (myApplication.getResources()," ",false);
  myParser.parse(resourceID);
  ArrayList<Number3d> myP=myParser.getVertices();
  ArrayList<Uv> myT=myParser.getTexCoords();
  ArrayList<Number3d> myN=myParser.getNormals();
```

Als nächste Aufgabe müssen die angelieferten Informationen in Arrays umgewandelt werden, die den Umwandlungsprozess in die für OpenGL verarbeitbaren Pufferstrukturen durchlaufen können. Im Fall der Texturkoordinaten, der Positionsdaten und der hier aus Platzgründen nicht abgedruckten Normalen beschränkt sich diese Operation auf das Hin- und Herkopieren der Daten:

```
float[] pos=new float[myP.size()*3];
for(int i=0;i<myP.size();i++)
{
  pos[i*3+0]=myP.get(i).x;
  pos[i*3+1]=myP.get(i).y;
  pos[i*3+2]=
  myP.get(i).z;
}

float[] tex = new
float[myT.size()*2];
for(int i=0;
i<myT.size();i++)
{
  tex[i*2+0]=
  myT.get(i).u;
  tex[i*2+1]=
  myT.get(i).v;
}
```



Die OBJ-Datei wurde erfolgreich in die Renderingpipeline geladen – doch es ist etwas faul (Bild 5)

Interessant ist an dieser Stelle die Berechnung der Koordinaten. Die Laufvariable *i* enumeriert die Quellelemente. Um in das Zielkoordinatensystem zu gelangen, ist eine Multiplikation mit dem Faktor zwei beziehungsweise drei erforderlich.

Da manche Shader das Vorhandensein von Farbdaten voraussetzen, erzeugen wir ein leeres Array mit Materialfarben:

```
float[] col=new float[myN.size()*4];
for(int i=0;i<myN.size();i++)
{
  col[i*4+0]=0;
  col[i*4+1]=0;
  col[i*4+2]=0;
  col[i*4+3]=0;
}
```

An dieser Stelle könnte ein Aufruf des für die Erzeugung von vier Arrays zuständigen Konstruktors folgen, der die Instanz der Modellklasse einsatzbereit machen würde. Leider ist dem in der Praxis nicht so: Aufrufe von *this()* und *super()* müssen in Java immer als erster Aufruf der jeweiligen Methode stehen.

Zur Umgehung dieses Problems zerlegen wir den bestehenden Konstruktor in eine Untermethode, was zur folgenden finalen Konstruktor- und Methodenhierarchie führt:

```
public NMGModelClass(float[] _pos, float[] _col,
float[] _nor, float[] _tex) {
  setupFromArray(_pos,_col,_nor, _tex);
}
public NMGModelClass(Application myApplication,
int resourceID) {
  ...
  setupFromArray(pos, col, nor, tex);
}
void setupFromArray(float[] _pos, float[] _col,
float[] _nor, float[] _tex)
```

Tabelle 1: Darstellungsmethoden

GL_POINTS	Zeichnet beliebige Punkte auf den Bildschirm. Jeder Vertex beschreibt einen alleinstehenden Punkt im Raum.
GL_LINES	Zeichnet isolierte Linien auf den Bildschirm. Je zwei Vertices beschreiben ein Liniensegment.
GL_LINE_STRIP	Zeichnet eine Gruppe von miteinander verbundenen Linien.
GL_LINE_LOOP	Zeichnet eine Gruppe von Linien, wobei das Ende der letzten mit dem Beginn verbunden wird.
GL_TRIANGLES	Zeichnet isolierte Dreiecke auf den Bildschirm. Je drei Vertices beschreiben ein Dreieck.
GL_TRIANGLE_STRIP	Zeichnet miteinander verbundene Dreiecke. Bei Übergabe von 1,2,3,4,5,... entstehen die Dreiecke 1–2–3, 2–3–4, usw.
GL_TRIANGLE_FAN	Zeichnet einen Dreiecksventillator. Der erste Vertex beschreibt den Ausgangspunkt, die weiteren Dreiecke sind dann 1–2–3, 1–3–4, 1–4–5 usw.

```
{
  myPosData=_pos;
}
```

Hier zeigen sich die Vorteile unserer Modellklasse zum ersten Mal. Die Implementierung des Affen beschränkt sich auf zwei kleine Anpassungen im Renderer. Das Modellfeld wird um einen Eintrag größer gestaltet, der nach der Erzeugung der Würfel mit den betreffenden Informationen befüllt wird:

```
@Override
public void onSurfaceCreated(GL10 gl,EGLConfig config) {
  for(int i=0;i<10;i++) {...}
  myModelCollection[10]=new NMGModelClass
  (myApplication, R.raw.affemittextur);
  myModelCollection[10].dx=0;
  myModelCollection[10].dy=0;
  myModelCollection[10].dz=0;
  myModelCollection[10].setNewPos(-5,-5,-5);
```

Damit ist auch diese Variante des Codes zur Ausführung bereit. Da sich der Affe in der Mitte des Bildschirms befindet und sich nicht bewegt, sollten Sie nach dem Start eine an Bild 5 erinnernde Szene zu sehen bekommen.

Eine Frage der Sequenzierung

Unsere von Hand zusammengebastelten Modelle bestanden aus einer Abfolge von Dreiecken. In OpenGL ES wird dies der Methode *drawArrays* durch Übergeben der Konstante *GL_TRIANGLES* mitgeteilt. Findet er sich in der Parameterliste, so wird das übergebene Array als Liste nicht zusammenhängender Dreiecke betrachtet:

```
GLES20.glDrawArrays(GLES20.GL_TRIANGLES, 0,
myModelCollection[i].countTriangles());
```

Diese Art der Modellbeschreibung ist einfach zu verstehen, führt aber zu massiven Redundanzen. Die Beschreibung eines aus nur zwei Dreiecken bestehenden Elements setzt sechs Punkte voraus. In vielen Fällen ist der Gutteil der abzuspeichernden Punkte redundant.

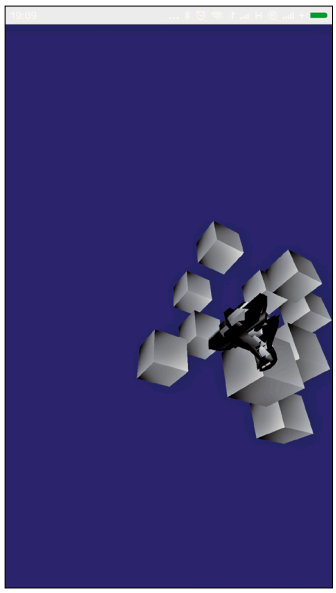
Dies ist nur eine von vielen möglichen Darstellungsmethoden. OpenGL ES unterstützt unter Android die in Tabelle 1 vorgestellten Versionen, die durch im GLES20-Objekt angelegte Konstanten aktiviert werden. Unser von Min3D bereitgestelltes Objekt liefert mit *GL_TRIANGLE_STRIP* die besten Resultate.

Da unsere Eingine zwecks höherer Flexibilität mit verschiedenen Darstellungsarten umgehen können soll, lösen wir das Problem durch Einführung einer Getter-Methode:

```
public int
getOGLDrawMode() {
  return myDrawMode;
}
```

Je nach Art der Initialisierung setzen wir den Inhalt des Flags auf die jeweilige Konstante. OpenGL ES kommt uns an dieser Stelle insofern entgegen, als die Konstanten in Form von Integers vorliegen:

```
public NMGModelClass
(float[] _pos,
float[] _col,
float[] _nor,
float[] _tex) {
  myDrawMode = GLES20.
GL_TRIANGLES;
}
public NMGModelClass
(Application
myApplication, int
resourceID) {
  myDrawMode = GLES20.GL_TRIANGLE_STRIP;
```



Eine Änderung des Zeichenmodus sorgt für besseres Aussehen (Bild 6)

Nicht nur bei der Programmierung von Spiele-Engines ist es extrem hilfreich, so viel Logik wie irgendwie möglich in Konstruktoren auszulagern. Wer die Nutzer seiner Klassen zu manuellen Handlungen animiert, muss über kurz oder lang mit technischen Problemen rechnen.

Im Renderer erfolgt der Aufruf von *glDrawArrays* nun mit dem Resultat dieses Aufrufs:

```
public void onDrawFrame(GL10 gl)
{
  for(int i=0;i<myModelCollection.length;i++)
  {
    ...
    GLES20.glDrawArrays
    (myModelCollection[i].getOGLDrawMode(), 0,
    myModelCollection[i].countTriangles());
```

▶

Listing 3: affewithtextur.obj

```
# Blender v2.69 (sub 0) OBJ File: ' '
# www.blender.org
o Suzanne
v 0.437500 0.164062 0.765625
v -0.437500 0.164062 0.765625
v 0.500000 0.093750 0.687500
...
vn -0.247398 -0.923066 -0.294522
vn 0.247398 -0.923066 -0.294522
vn 0.000000 0.000000 1.000000
...
f 47/1/1 45/1/1 3/1/1 1/1/1
f 4/1/2 46/1/2 48/1/2 2/1/2
f 45/1/3 43/1/3 5/1/3 3/1/3
```

```
    }
}
```

Damit ist das Programm fürs Erste fertig. Bild 6 zeigt, was Sie sich erwarten dürfen.

Auswahl des Renderingmodus

Auch wenn wir durch Auswahl des richtigen Renderingmodus weitergekommen sind – wie in Blender sieht der Affe nach wie vor nicht aus. Dies liegt an einer Besonderheit des OBJ-Formats: Die Vertices werden im zweiten Schritt zu Faces zusammengefasst, die die eigentlichen Flächen beschreiben. Dies lässt sich anhand eines kleinen Snippets aus der Datei *affewithtextur.obj* demonstrieren (Listing 3).

Die mit *v, vn* und *vt* beginnenden Zeilen definieren Positions-, Textur- und Normalkoordinaten. Darunter findet sich die Deklaration der Flächen, die in OpenGL als Faces bezeichnet werden.

Jedes Face ist durch drei oder vier Punkte beschrieben, die aus je einem Vertex-, einem Textur- und einem Normalindex bestehen. Ein Vertex kann so in mehreren Flächen wiederverwendet werden, was in der Praxis zu wesentlichen Speichereinsparungen führt.

	i1	i2	i3	i4
j1	x	1	1	1
j2	1	x	2	2
j3	1	2	x	3
j4	1	2	3	x

Wer A mit B vergleicht, erspart sich den Vergleich von B mit A (Bild 7)

OBJ-Dateien enthalten oft – wie in den zwei folgenden Zeilen gezeigt – vierwertige Faces, die zwei zusammenhängende Dreiecke beschreiben:

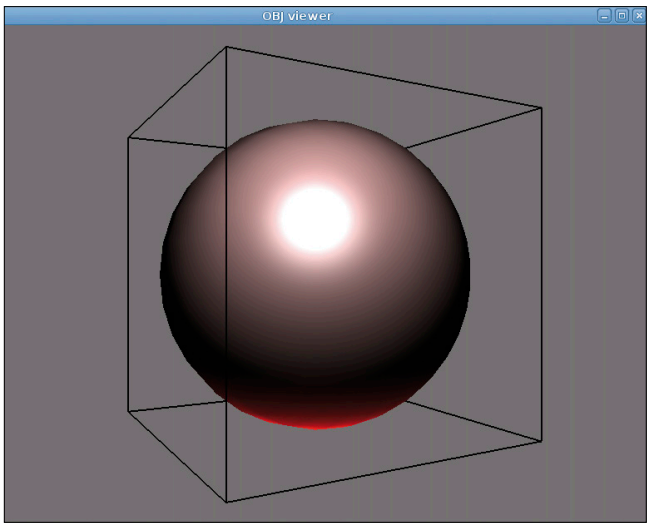
```
f 557/190/548 555/191/548 509/192/548 573/193/548
f 568/194/549 572/195/549 556/196/549
```

Die unter www.martinreddy.net/gfx/3d/OBJ.spec bereitstehende Spezifikation liefert weitere Informationen über die Rolle der einzelnen Elemente einer OBJ-Datei.

Indizierte Modelle

OpenGL kann indizierte Modelle auf Wunsch direkt rendern. Da die in Min3D implementierte Pipeline beim Parsen von Faces die Textur- und Normaleninformationen unterschlägt, wollen wir uns mit dem bisher Erreichten zufriedengeben und stattdessen andere Konzepte betrachten.

In praktischem Code wäre es sinnvoll, die in Min3D teilweise implementierte Parsinglogik zu erweitern. Da aus diskreten Arrays aufgebaute Modelle dort nur wenig Bedeutung haben, ist die Nutzung einer komplett indirekten Pipeline mitunter ebenfalls effektiver. Auch wenn Affe und Würfel am Bildschirm sind: Wirklich realistisch ist unsere Szene nicht.



Die Bounding Box umschließt das Objekt (Bild 8)

Dies liegt daran, dass die Modelle nach Belieben umherfliegen – neben dem Verlassen des sichtbaren Bereichs können die einzelnen Objekte auch ineinandergleiten.

Um das Verschwindens der Modelle zu beheben, müssen wir den Bewegungen Grenzen einschreiben. Das In-der-Szene-Halten ist vergleichsweise einfach:

```
public void update()
{
  setNewPos(xpos+dx, ypos+dy, zpos + dz);
  if(xpos < -5 ||xpos > 5 )dx*=-1;
  if(ypos < -5 ||ypos > 5 )dy*=-1;
```

Listing 4: generateBounds()

```
void generateBounds()
{
  myMinx=myMaxx=myPosData[0];
  myMiny=myMaxy=myPosData[1];
  myMinz=myMaxz=myPosData[2];
  int triCountCache=countTriangles();
  for(int i=1;i<triCountCache;i++) {
    if (myPosData[i * 3 + 0] > myMaxx)
    myMaxx = myPosData[i * 3 + 0];
    if (myPosData[i * 3 + 0] < myMinx)
    myMinx = myPosData[i * 3 + 0];
    if (myPosData[i * 3 + 1] > myMaxy)
    myMaxy = myPosData[i * 3 + 1];
    if (myPosData[i * 3 + 1] < myMiny)
    myMiny = myPosData[i * 3 + 1];
    if (myPosData[i * 3 + 2] > myMaxz)
    myMaxz = myPosData[i * 3 + 2];
    if (myPosData[i * 3 + 2] < myMinz)
    myMinz = myPosData[i * 3 + 2];
  }
}
```

```
  if(zpos < -5 ||zpos > 5 )dz*=-1;
}
```

Nach jeder Aktualisierung folgt eine Prüfung der Position in den drei Richtungen. Ist eine davon nicht mehr legal, so wird die jeweilige Bewegungsrichtung umgekehrt. Beim nächsten Frame bewegt sich das Objekt so automatisch in die legale Zone zurück, weshalb Sie den abstrakten Kugelspielen nun länger zusehen können.

Die Erfassung von Kollisionen zweier Objekte ist schon wegen der Vielzahl der involvierten Punkte schwieriger. Wer jeden Vertex eines 200 Seiten schweren Objekts mit jedem Vertex eines 36 Vertices umfassenden Objekt vergleicht, provoziert bei unintelligenter Implementierung 200*36 Vergleichsoperationen. Die Nutzung des in Bild 7 gezeigten Verfahrens reduziert diese zwar, das Resultat ist für Echtzeitanwendungen trotzdem nicht tragbar.

Als Alternative dazu hat sich das Verfahren der Bounding Boxes etabliert. Es handelt sich dabei um oktagonale Hilfsobjekte, die um die eigentliche Objektgeometrie errichtet werden. Bild 8 illustriert die Vorgehensweise anhand einer Kugel, deren Ränder die Größe des Begrenzungselements beschreiben. Für Entwickler ist diese Vorgehensweise hilfreich, weil der Vergleich zweier Bounding Boxes mit vergleichsweise wenig Rechenleistung auskommt. Eine genauere Kollisionsprüfung muss, wenn überhaupt, nur bei wenigen und vielversprechenden Konfliktkandidaten erfolgen.

Leider bietet OpenGL selbst kein API zur Erzeugung einer Bounding Box an. Da jeder Entwickler seine Koordinaten auf eine andere Art und Weise speichert, würde das Einführen einer Berechnungsfunktion in massive Arbeit ausarten.

Erfreulicherweise ist die Behebung dieses Missstands nicht sonderlich schwer. Wir gehen in den folgenden Schritten nicht einmal davon aus, dass alle zu verarbeitenden Objekte um den Mittelpunkt, also 0|0|0, aufgebaut sind.

Die minimale Ausbreitung einer Bounding Box ist stets durch den ersten Objektvertex gegeben. Am Anfang dient er sowohl als Unter- als auch als Obergrenze. Unser Errichtungsalgorithmus durchläuft im nächsten Schritt alle Vertices des Objekts. Ist eine der Koordinaten größer oder kleiner als die momentanen Außengrenzen der *BoundingBox*, so wird die jeweilige Grenze aktualisiert (Listing 4).

triCountCache dient während der Ausführung der Berechnungsmethode als Cache. Die *size()*- beziehungsweise *count()*-Operation von *Collections* erweist sich in der Praxis insbesondere in engen Schleifen als Performancefresser erster Güte. An besonders hoher Geschwindigkeit interessierte Entwickler könnten die Multiplikation *i*3* durch Einführung einer weiteren Puffervariable einsparen.

Für die effizientestmögliche Einbindung rufen wir *generateBounds()* direkt aus dem Konstruktor der Modellklasse auf, um so schließlich mit einer vollwertigen Bounding Box zu starten.

Grenze in Bewegung

Die Berechnung einer stationären Bounding Box ist nur ein Teil der Lösung. Modelle haben die unangenehme Eigenschaft, ihre Position während der Ausführung des Spiels zu verändern. Die weiter oben genannte Translationsmatrix ermöglicht neben Skalierungen und Verschiebungen auch die Rotation. Im Moment wollen wir uns auf die ersten beiden Bewegungsarten beschränken.

Listing 5: transformBB

```
public float[] transformBB() {
  float[] lowerPoint=new float[4];
  lowerPoint[0]=myMinx;
  lowerPoint[1]=myMiny;
  lowerPoint[2]=myMinz;
  lowerPoint[3]=1;
  float[] higherPoint=new float[4];
  higherPoint[0]=myMaxx;
  higherPoint[1]=myMaxy;
  higherPoint[2]=myMaxz;
  higherPoint[3]=1;
  float[] resultPoints=new float[7];
  Matrix.multiplyMV(resultPoints,0,myNoRotMatrix,0,
  lowerPoint,0);
  Matrix.multiplyMV(resultPoints,3,myNoRotMatrix,0,
  higherPoint,0);
  return resultPoints;
}
```

Da das manuelle Verschieben von Objekten in Arbeit ausartet, nutzen wir stattdessen eine um Rotationsinformationen erleichterte Version der normalen Weltmatrix. Sie wird im Rahmen von *setNewPos* vor dem Einschreiben der Rotation generiert und wartet in einer Membervariable der Objektklasse:

```
float[] myNoRotMatrix;
public void setNewPos(float _x, float _y, float _z) {
  xpos=_x;
  ypos =_y;
  zpos=_z;
  Matrix.setIdentityM(myModelMatrix, 0);
  Matrix.translateM(myModelMatrix, 0, xpos, ypos, zpos);
  myNoRotMatrix=myModelMatrix.clone();
```

Die in *myNoRotMatrix* befindlichen Transformationsdaten erlauben uns nun das Verschieben der beiden Eckpunkte der Quader. In unserem Beispiel findet sich die dazu notwendige Logik in einer dedizierten Methode, die nach Bedarf aufgerufen werden kann (Listing 5).

transformBB liefert ein Array aus Float-Werten zurück, in dem die beiden Begrenzungspunkte übereinanderliegend gespeichert werden. Die in Androids OpenGL-Unterstützungsbibliothek enthaltene Methode *multiplyMV* nimmt sowohl zu ihren Quell- als auch zu ihren Zielparametern Offsets entgegen, die das Ausblenden von nicht benötigten Teilen des Feldes erlauben. Durch Nutzen von *3* weisen wir den zweiten Aufruf von *multiplyMV* dazu an, seine Arbeit erst ▶

im dritten Feld des Arrays *resultPoints* zu beginnen. Aufmerksame Leser bemängeln an dieser Stelle, dass die beiden Punktarrays vierwertig sind; das Resultatsarray ist statt sechs sieben Floats hoch. Dies liegt daran, dass die Translationsmatrix 4 x 4 Elemente enthält. Wäre der zu multiplizierende Vektor nicht ebenfalls vier Elemente hoch, so käme es zu einem Fehler. Der hier nicht vorhandene Achswert wird immer als *1* angenommen.

Da wir die BoundingBox in jedem Durchlauf von *update()* benötigen, aktualisieren wir sie im Rahmen der Abarbeitung von *setNewPos*:

```
public void setNewPos(float _x, float _y, float _z) {
  myBB=transformBB();
}
```

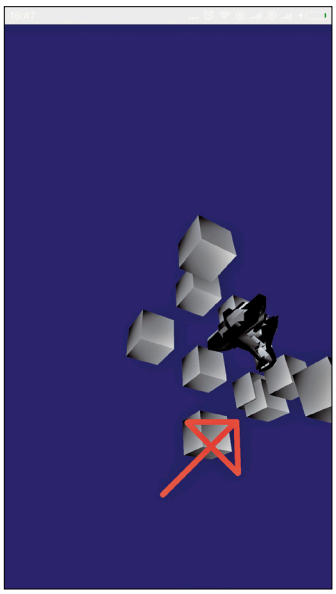
Nachdem wir zur Transformation der Koordinaten fähig sind, können wir uns der eigentlichen Realisierung der Kollisionserkennung zuwenden. Die Methode zur Beschränkung der Durchlaufindizes wird hier durch Anpassung der Startwerte der beiden *For*-Schleifen realisiert:

```
@Override
public void onDrawFrame(GL10 gl) {
  for (int i = 0; i < myModelCollection.length; i++) {
    myModelCollection[i].update();
  }
  for(int i=0; i < myModelCollection.length; i++){
    for(int j=i+1;j < myModelCollection.length; j++){
      if(compareBBs(myModelCollection[i].myBB,
      myModelCollection[j].myBB)==true)
      {       }
    }
  }
```

Der eigentliche Vergleich findet sich in der Methode *compareBBs*. Vergleiche von Kollisionen zwischen Rechtecken

Listing 6: Geschwindigkeitsvektoren tauschen

```
if(compareBBs(myModelCollection[i].myBB,
myModelCollection[j].myBB)) {
  float cX,cY,cZ;
  cX=myModelCollection[i].dx;
  cY=myModelCollection[i].dy;
  cZ=myModelCollection[i].dz;
  myModelCollection[i].dx=myModelCollection[j].dx;
  myModelCollection[i].dy=myModelCollection[j].dy;
  myModelCollection[i].dz=myModelCollection[j].dz;
  myModelCollection[j].dx=cX;
  myModelCollection[j].dy=cY;
  myModelCollection[j].dz=cZ;
  myModelCollection[j].update();
  myModelCollection[i].update();
}
```



Zwei der Quader kleben zusammen (Bild 9)

lassen sich am Einfachsten lösen, wenn man sie auf die Betrachtung von Einzelachsen reduziert. Berührt sich ein Element im Bereich der x-, der y- und der z-Achse, so hat es auch im dreidimensionalen Raum Kontakt:

```
public boolean
compareBBs(float[] _
bb1,float[] _bb2) {
  boolean touchX=false,
  touchY=false,
  touchZ=false;
```

Die Prüfung auf Achsebene ist vergleichsweise einfach. Wir prüfen, ob sich entweder der Anfangs- oder der Endpunkt der Bounding Box des ersten Objekts innerhalb der Strecke befindet, die von der zweiten Bounding Box auf der jeweiligen Achse belegt wird:

```
  if(_bb1[0]>_bb2[0] && _bb1[0]<_bb2[3])touchX=true;
  if(_bb1[3]>_bb2[0] && _bb1[3]<_bb2[3])touchX=true;
```

Die Arbeit mit den restlichen Achsen erfolgt dann analog. Am Ende der Methode vergleichen wir die *touch*-Werte. *true* wird nur dann zurückgegeben, wenn alle drei *true* sind:

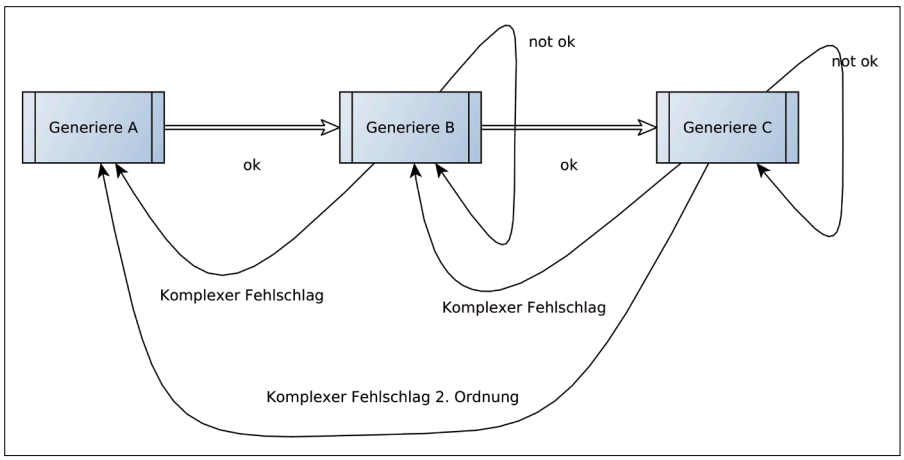
```
  if(_bb1[1]>_bb2[1] && _bb1[1]<_bb2[4])touchY=true;
  if(_bb1[4]>_bb2[1] && _bb1[4]<_bb2[4])touchY=true;
  if(_bb1[2]>_bb2[2] && _bb1[2]<_bb2[5])touchZ=true;
  if(_bb1[5]>_bb2[2] && _bb1[5]<_bb2[5])touchZ=true;
  return (touchX && touchY && touchZ);
}
```

Die genaue Berechnung der Geschwindigkeiten von zwei Objekten nach einer Kollision ist aufwendig. Als physikalische Grundlagen dienen Impuls- und Energieerhaltungssatz. Für die Praxis reicht es oft aus, wenn die beiden Elemente ihre Geschwindigkeitsvektoren wie in Listing 6 tauschen.

Wer das Programm im vorliegenden Zustand ausführt, sieht nach einiger Zeit das in Bild 9 gezeigte Verhalten. Es dürfte bei funktionierender Kollisionserkennung eigentlich nicht vorkommen, tritt hier aber aufgrund geschickter Parametrierung des Quader-Generationsalgorithmus auf.

Per Zufallsgenerator erzeugte Spielelemente sind – wie sollte es auch anders sein – zufällig verteilt. Es kann dabei durchaus vorkommen, dass zwei Objekte im selben Platz landen: bei schlecht konstruierten Engines kann dies zum Komplettausfall der Kollisionserkennung führen.

Als einfache Lösung bietet sich das sofortige Überprüfen der neu erstellten Quader an. Ist sie nicht auf einem legalen



Backtracking geht bei Bedarf einen oder gar mehrere Schritte zurück (Bild 10)

Ort, so wird die Position neu berechnet. Leider ist auch dies nicht allumfassend, da sich ein derartiger Generator in eine Todeszone rechnen kann.

Wer das Problem der Generation von Levels vollumfänglich lösen möchte, sollte auf das in Bild 10 illustrierte Backtracking-Verfahren setzen. Die Engine baut dabei fürs Erste drauflos. Wenn es nach einigen Retries an einer bestimmten Stelle nicht weitergeht, wird der vorhergehende Würfel eliminiert. Tritt danach abermals ein Totpunkt auf, so geht die Engine einen weiteren Schritt zurück.

Alternativ dazu ist es auch möglich, die Generierung nach einer gewissen Anzahl von Fehlschlägen zu beenden – ob der Nutzer mit dem erzeugten Level dann zufrieden ist, ist allerdings von Situation zu Situation unterschiedlich.

Unsere Bounding Boxes werden im englischsprachigen Raum als AABB bezeichnet. Hinter dem Begriff versteckt sich das Akronym Axis Aligned Bounding Box. Es handelt sich dabei um ein spezielles Rechteck, dessen Kanten parallel zu den Achsen stehen – bei der Berechnung der Kollisionswahrscheinlichkeit ist dies eine nicht unerhebliche Erleichterung.

Zur Lösung dieses Problems gibt es mehrere Methoden. Die aus mathematischer Sicht schwierigste ist, den Schnitt zwischen zwei beliebigen im Raum schwebenden Quadern zu beschreiben. Der Vergleich aller Vertices ist aus Rechenleistungssicht am teuersten, setzt aber am wenigsten Code voraus. Ein Mittelding zwischen beiden ist das von Jeff Lander im Jahr 2001 bei GamaSutra beschriebene Verfahren (www.gamasutra.com/view/feature/131833/when_two_hearts_collide_.php?print=1), das durch Rotation der schon vorhandenen Bounding Box eine Gruppe von zu maximierenden Koordinaten erzeugt.

Hintergrundmusik ist wichtig

Heutige Computerspiele kommen nicht ohne Soundeffekte aus. Bei Titeln mit ausreichendem Budget arbeiten oft mehrere Komponisten an der Hintergrundmusik. Der US-amerikanische Komponist Kevin MacLeod hat sich im Kreis kleinerer und mittlerer Entwickler mit seinem Studio IncompeTech Kultstatus erarbeitet. Die Website https://incompetech.com/music/royalty-free enthält eine Vielzahl verschiedener Musikstücke, die sich gegen Namensnennung kostenfrei verwenden lassen. Unser Beispiel nutzt den Track Professor Umlaut, der unter https://www.youtube.com/watch?v=Oel6rxdIQag beworben wird. Fügen Sie die Datei in den Raw-Ressourcenordner ein, um sie im Rahmen des Deployments der Applikation in Richtung Smartphone oder Tablet zu schicken.

Hintergrundmusik wird in Android am Besten über eine Instanz von MediaPlayer verwaltet: Die Klasse bietet mit *setVolume* eine Funktion an, mit der sich die Lautstärke der beiden Kanäle flexibel einstellen lässt:

```
public void playMusic() {
  myBGM = new MediaPlayer();
  myBGM.setVolume(0.5f, 0.5f);
```

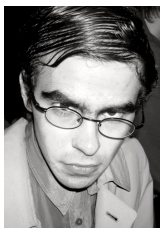
Im nächsten Schritt bevölkern wir die Klasse mit den Inhalten der MP3-Datei und geben Sie zum Abspielen frei:

```
  AssetFileDescriptor afd;
  try {
    afd = myApplication.getResources().
    openRawResourceFd(R.raw.professorumlaut);
    myBGM.setDataSource(afd.getFileDescriptor(),
    afd.getStartOffset(), afd.getLength());
    myBGM.setLooping(true);
    myBGM.prepare();
  } catch (Exception e1) {
    e1.printStackTrace();
  }
  myBGM.start();
}
```

Damit ist auch dieser Teil des Programms fertig. Schicken Sie es auf Ihr Telefon, um sich an der Musik zu erfreuen.

Fazit

Spieleentwicklung auf Basis von OpenGL ES ist ein weites Feld. Zu künstlicher Intelligenz, der Generierung von Levels und der Erzeugung fortschrittlicher Grafikeffekte ließen sich Bücher schreiben. Unsere Reise durch die Welt der Games endet schon aus Platzgründen an dieser Stelle. ■



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

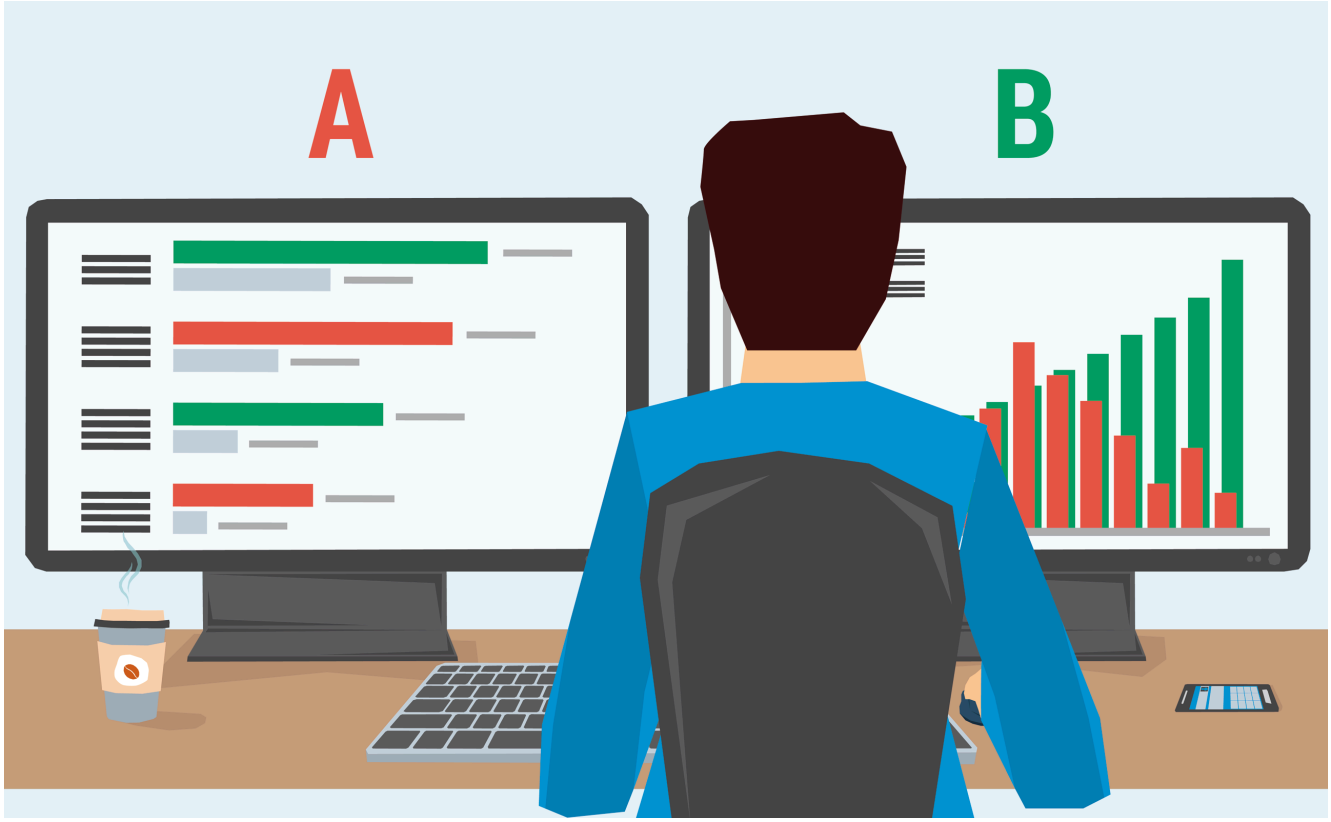


Bild: Shutterstock / irin73bal

ERSTELLEN EINER TABLE-VIEW MIT SPALTEN

Tabelle mit Spalten

So erstellt man eine eigene Table-View-Klasse, die neben Zeilen auch Spalten bietet.

Die Klasse *UITableView* gehört wohl zusammen mit ihrem View-Controller *UITableViewController* zu den am häufigsten verwendeten Klassen in der iOS-Entwicklung. Dabei ist der Name eigentlich ein wenig irreführend: Eine UITableView listet zwar einzelne Zellen in einer vertikalen Reihe auf, das war es aber auch schon. Spalten – wie man sie typischerweise von Tabellen kennt – sind mit einer UITableView von Haus aus nicht realisierbar.

Während dieser Umstand bei Einführung des Ur-iPhones noch vernachlässigbar war – der kleine Bildschirm eignete sich kaum für Tabellenansichten mit mehreren Spalten – ist das heute inzwischen anders. Das iPad hat zwischenzeitlich die Bühne betreten und mit der Einführung des 12,9 Zoll großen iPad Pro Familienzuwachs erhalten, der mehr Bildschirmplatz bietet als so manches MacBook. Die Umsetzung einer Tabellenansicht mit mehreren Spalten ergibt gerade bei dieser Gerätegattung in entsprechenden Fällen Sinn und ließe sich dank der großen Bildschirme auch tatsächlich realisieren und sinnvoll nutzen.

Doch nicht mit *UITableView*; diese Klasse gibt das gewünschte Verhalten wie bereits beschrieben einfach nicht her. Wer also eine Tabelle mit Spalten in eigenen iOS-Apps benötigt, muss selbst Hand anlegen. Das tat auch ich im Zuge eines Kundenprojekts, wo ich genau solch eine Tabelle mit mehreren Zellen und Spalten benötigte. Und um in Zukunft nicht immer damit bei null anfangen zu müssen, achtete ich darauf, diese View mitsamt all ihrer Bestandteile möglichst generisch zu programmieren, um sie – ähnlich wie eine *UITableView* – wiederverwendbar zu machen. In diesem Artikel beschreibe ich mein Vorgehen bei der Programmierung dieser eigenen Table-View, darüber hinaus steht der gesamte Code auf GitHub zum Download bereit.

Basis der eigenen Table-View: Delegation

Im ersten Schritt der Entwicklung der beschriebenen View steht die Frage, wie das Interface aussehen soll und wie unsere neue eigene Table-View-Klasse genutzt und konfiguriert werden kann. Irgendwie müssen wir sie ja schließlich mit allen notwendigen Informationen füttern, damit sie sich immer dynamisch unseren Wünschen entsprechend aufbaut.

Wie auch bei der Original-Klasse *UITableView* ist für unsere eigene, mehrspaltige Table-View Delegation eine essenzi-

elle Grundlage. Warum? Weil wir unsere Table-View mit vielen verschiedenen Informationen füttern müssen: Wie viele Spalten soll sie anzeigen? Wie viele Reihen besitzt sie? Welche Inhalte werden in den verschiedenen Zellen angezeigt? Wie sollen die Header lauten? All das sind Fragen, die unsere neue generische View-Klasse selbst nicht beantworten kann (und soll), die sie aber abfragen und verarbeiten muss; somit ist sie ein perfekter Fall für Delegation (Bild 1).

Alternativ bliebe nur, die Klasse mit passenden Properties auszustatten, die man setzen kann und die entsprechend den Aufbau und das Aussehen unserer Tabelle regeln. Doch das wäre immens umständlich, aufwendig und unübersichtlich. Man stelle sich nur einmal vor, wie man auf diese Art und Weise verschiedene Header-Titel setzen oder die Inhalte der einzelnen Zellen festlegen soll. Dafür ist die geplante Klasse einfach zu mächtig und es ist sinnvoll, zumindest einzelne Aufgaben mittels Delegation auszulagern. Properties können dann immer noch für einzelne separate Informationen verwendet werden, die allgemein für eine gesamte Table-View gelten.

Die Sache mit dem Namen

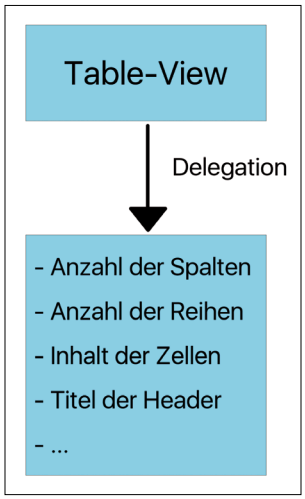
Bevor ich nun mit der eigentlichen Implementierung und Umsetzung unserer eigenen Table-View-Klasse beginne, möchte ich zuvor noch einen durchaus wichtigen Punkt klären: den Namen. Bisher spreche ich immer von »unserer eigenen Table-View«, doch natürlich ist das kein wirklich brauchbarer Name für eine entsprechende Klasse. *TableView* alleine ist ok, kann aber leicht zu Verwechslungen mit Apples *UITableView*-Klasse führen.

Ich habe einige Zeit darüber nachgedacht und leider keine wirklich zufriedenstellende Lösung für einen guten Namen gefunden. Ich dachte an etwas wie *MatrixView* oder auch *ListView*, fand diese Beispiele aber ebenfalls nicht wirklich passend. Schließlich geht es um eine Tabelle, und das wollte ich auch zum Ausdruck bringen.

Da ich diese Klasse als generische View schreiben wollte, die ich in beliebigen Projekten immer und immer wieder verwenden kann, entschloss ich mich letzten Endes dazu, einfach mein eigenes Framework-Kürzel voranzustellen. Somit habe ich die Klasse an meiner Stelle auf *TSTableView* getauft. Zwar besteht dabei weiterhin durchaus Verwechslungsgefahr mit *UITableView*, doch das Präfix differenziert an dieser Stelle für meine Zwecke gut genug, ohne den Sinn und Zweck unserer neuen Klasse zu verwässern. Selbstverständlich dürfen Sie bei der Entwicklung der Klasse jeden Namen verwenden, der Ihnen sinnvoll und passend erscheint.

Die Basistypen

Bevor wir mit der Umsetzung der eigentlichen *TSTableView*-Klasse beginnen, möchte ich Ihr Augenmerk zuvor noch auf eine andere Frage lenken: Wie bilden wir einzelne Zellen innerhalb unserer Tabelle am besten ab? Bei einer *UITableView* ist das relativ offensichtlich: Es gibt einzelne Reihen von Zellen, die nacheinander angezeigt werden, die Reihen sind also somit das ausschlaggebende Kriterium.



Delegation erlaubt es der Table-View, verschiedenste Informationen aus anderen Quellen zu beziehen (Bild 1)

Header 0	Header 1	Header 2
Reihe 0 | Spalte 0	Reihe 0 | Spalte 1	Reihe 0 | Spalte 2
Reihe 1 | Spalte 0	Reihe 1 | Spalte 1	Reihe 1 | Spalte 2
Reihe 2 | Spalte 0	Reihe 2 | Spalte 1	Reihe 2 | Spalte 2

Die Zellen setzen sich aus ihrer zugehörigen Reihe und Spalte zusammen (Bild 2)

Doch bei unserer Table-View ist das nicht so. Die Reihe ist nur der eine Teil der Beschreibung einer Zelle. Da unsere Table-View auch über mehrere Spalte verfügen kann, definiert sich eine Zelle in diesem Fall aus einer Kombination aus Reihe und Spalte (Bild 2).

Diese Zusammensetzung der einzelnen Zellen unserer Table-View müssen wir sinnigerweise bereits im Vorfeld passend abbilden; schließlich sind sie eine essenzielle Grundlage unserer Table-View. Zu diesem Zweck können wir am einfachsten eine kleine Structure erstellen, die über genau zwei Eigenschaften verfügt: einen Index für eine Reihe einer Tabelle sowie einen Index für die Spalte einer Tabelle. Instanzen einer solchen Structure enthalten dann genau die Informationen, die wir benötigen, um Zellen eindeutig zu identifizieren und darüber hinaus auf verschiedene Reihen und Spalten zuzugreifen.

Weitere Optimierung

In diesem Zuge können wir auch gleich eine weitere Optimierung vornehmen, die die Arbeit mit den verschiedenen Indexen erleichtert und übersichtlicher gestaltet. Standardmäßig würden wir die genannten Indexe als einfache Integer definieren, da sie darüber ja die eindeutige Position einer Zelle festlegen. Schöner ist es aber, einen eigenen Typ zu verwenden, wenn explizit mit den Reihen- und Spalten-Indexen unserer Table-View gearbeitet wird. Damit macht man deutlich, dass es sich hier nicht um irgendwelche Integer handelt, sondern dass diese sich auf die Position eines Elements der Table-View beziehen.

Um das entsprechend so umzusetzen, lässt sich in Swift ein einfacher sogenannter Typealias erstellen, dem wir einen passenden Namen geben und der dann einem bereits vorhandenen Typ entspricht, dem wir eben diesem Typealias zuweisen.

Das folgende Listing zeigt die eben genannte Umsetzung einer Structure zur Abbildung der Position einer Zelle sowie die Definition von Typealiassen für die Indexe einer Reihe und einer Spalte der Table-View. Die Structure habe ich – passend zu meinem ver- ▶

wendeten Table-View-Präfix TS – *TSTableViewCellCoordinate* genannt; zur Abbildung der Koordinaten einer Zelle innerhalb der Tabelle:

```
typealias TSTableViewColumnIndex = Int
typealias TSTableViewRowIndex = Int
struct TSTableViewCellCoordinate {
  var columnIndex: TSTableViewColumnIndex
  var rowIndex: TSTableViewRowIndex
}
```

Betrachten wir als Nächstes einmal den grundlegenden Aufbau des eigentlichen Herzstücks dieses Projekts: der *TSTableView*-Klasse. Diese ist dafür verantwortlich, eine Tabelle auf dem Bildschirm zu zeichnen und darzustellen. Dabei soll sie ein passendes Delegate-Objekt verwenden, das Einzelheiten und Details zum gewünschten Aufbau der Tabelle enthält. Grundlage dieses Delegates soll ein ebenfalls neu zu erstellendes Protokoll namens *TSTableViewDelegate* sein.

Dabei verzichte ich in diesem Projekt ganz bewusst auf die Verwendung zweier verschiedener Delegates, wie es beispielsweise bei der Klasse *UITableView* der Fall ist. Diese verfügt über zwei Delegate-Objekte – *dataSource* und *delegate* – die jeweils zu einem anderen Protokoll – einmal *UITableViewDataSource*, einmal *UITableViewDelegate* – konform sind. Damit entschlackt Apple die genannten Protokolle und teilt die Funktionen auf diese beiden auf.

Damit unsere Table-View ganz grundlegend überhaupt funktionieren kann, benötigt sie zwei essenzielle Informationen: Wie viele Reihen und wie viele Spalten sollen angezeigt werden? Nur anhand dieser Informationen können wir die Table-View korrekt erstellen und auf dem Bildschirm zeichnen. Und der Delegate der Table-View ist prädestiniert dafür, eben jene Fragen zu beantworten.

Daher können wir im ersten Schritt die Umsetzung unserer Table-View wie folgt beginnen: Wir erstellen das zugrunde liegende *TSTableViewDelegate*-Protokoll und deklarieren darin zwei Methoden – *numberOfRowsInTableView:* und *numberOfColumnsInTableView:* –, über die wir die Anzahl der Reihen und Spalten einer Tabelle erhalten:

```
protocol TSTableViewDelegate {
  func numberOfRowsInTableView(tableView: TSTableView)
  -> Int
  func numberOfColumnsInTableView(tableView:
  TSTableView) -> Int
}
```

Möglicherweise fragt sich an dieser Stelle der ein oder andere, warum dafür nicht stattdessen einfache Getter-Properties verwendet werden, was ich an dieser Stelle auch gerne erklären möchte. Das Problem bei reinen Properties ist nämlich, dass sie keine Parameter übergeben können. Das würde bedeuten, dass ein Objekt, das konform zu unserem *TSTableViewDelegate*-Protokoll ist, immer nur die Informationen zur Anzahl der Reihen und Spalten für eine einzige Tabelle zurückliefern kann, oder alternativ eine eigene Logik zur Unterscheidung mehrerer Tabellen implementiert. Schließlich kann es ja sein, dass ein Delegate-Objekt mehr als nur eine Tabelle als Delegate verwaltet.

Da wir aber stattdessen direkt eine passende Methode für die benötigte Information deklarieren, können wir darüber auch das jeweilige *TSTableView*-Objekt an den Delegate zurückliefern. So kann ein Delegate innerhalb der Implementierung dieser Methoden prüfen, um welche Tabelle es sich handelt, um die korrekte Information zurückzugeben. Daher übergeben wir eine Instanz der Klasse *TSTableView* als Parameter der im Protokoll deklarierten Methoden.

Der zweite grundlegende Schritt besteht dann noch darin, unsere eigentlich wichtigste Klasse – die *TSTableView*-Klasse – zu erstellen und dieser eine *delegate*-Property vom Typ *TSTableViewDelegate* zu implementieren:

```
class TSTableView: UIScrollView {
  var tableViewDelegate: TSTableViewDelegate?
}
```

TSTableView ist dabei von der Klasse *UIScrollView* abgeleitet, um das grundlegende Scrollverhalten out of the Box zu

Listing 1: Funktionen zum Erstellen der Header

```
class TSTableView: UIScrollView {
  ...
  private let defaultTableViewCellIndention:
  CGFloat = 8
  private let defaultTableViewHeaderHeight:
  CGFloat = 44
  private var tableViewHeaderLabels = [UILabel]()
  ...
  private func createHeaderViews() {
    for index in 0 ..<
    tableViewDelegate!.numberOfColumnsInTableView
    (self) {
      createHeaderLabelForColumnAtIndex(index)
    }
  }
  private func createHeaderLabelForColumnAtIndex
  (index: TSTableViewColumnIndex) {
    let headerLabel = UILabel(frame: CGRectMake
    (xValueForColumnAtIndex(index) +
    defaultTableViewCellIndention, 0,
    widthForColumnAtIndex(index) -
    defaultTableViewCellIndention * 2,
    defaultTableViewHeaderHeight))
    headerLabel.font =
    UIFont.boldSystemFontOfSize(17)
    headerLabel.text = "Header \(index)"
    tableViewHeaderLabels.append(headerLabel)
  }
}
```

erhalten, ohne sich selbst um eine entsprechende Implementierung kümmern zu müssen.

Mit diesen grundlegenden Informationen gewappnet, können wir nun beginnen, die eigentliche Logik der *TSTableView*-Klasse zu implementieren.

Grundlagen von TSTableView

Bevor wir damit beginnen, Spalten und Reihen zu erstellen, ist es sinnvoll, einige grundlegende Funktionen zu implementieren, die für die korrekte Funktionsweise der Table-View notwendig sind. Dabei geht es zunächst einmal um die Breite und die x-Position jeder Spalte. Diese Informationen werden benötigt, um nacheinander verschiedene Spalten nebeneinander anzuzeigen. Die dafür notwendige Logik kann man idealerweise in zwei entsprechenden Methoden unterbringen, über die einmal die x-Position und einmal die Breite einer Spalte zurückgegeben wird. Als Parameter erwarten die beiden Methoden den Spaltenindex als *TSTableViewColumnIndex*-Objekt.

Für die Breite legen wir zunächst einen Defaultwert fest, den wir in einer entsprechenden Property *defaultTableViewColumnWidth* speichern. Damit ist die Methode zum Zurückliefern der Spaltenbreite noch sehr einfach; sie liefert einfach immer den Wert der genannten Property zurück. Später ergänzen wir hier noch eine Dynamik, um eine eigene Breite pro Spalte auch aus dem Delegate-Objekt heraus abzufragen und zu verwenden.

Die Methode zum Ermitteln der x-Position ist ein wenig komplexer. Zunächst definieren wir darin eine Variable mit dem Startwert *0*, der dem x-Wert der ersten Spalte entspricht. Anschließend werden in einer for-Schleife dann alle Spaltenindexe, beginnend beim Startwert *0* bis ausschließlich dem übergebenen Spaltenindex, durchlaufen. Das sorgt dafür, dass diese *for*-Schleife für jede Spalte durchlaufen wird, die sich vor der angeforderten Spalte befindet. Die angeforderte Spalte wird über den übergebenen Index-Parameter bestimmt.

Für jede Spalte, die auf diese Art und Weise durchlaufen wird, wird mittels der Methode zum Ermitteln der Breite einer Spalte eben jene Breite der jeweiligen Spalte berechnet. Diese Breite wird dann auf die Variable zum Speichern des x-Werts dazuaddiert. Daraus ergibt sich am Ende dann der x-Wert für die angefragte Spalte in Form des übergebenen Spaltenindex. Das nachfolgende Listing zeigt die bisherige vollständige Implementierung der *TSTableView*-Klasse mit den Methoden zur Ermittlung der x-Position und der Breite der Spalten der Table-View:

```
class TSTableView: UIScrollView {
  var tableViewDelegate: TSTableViewDelegate?
  private let defaultTableViewColumnWidth: CGFloat = 200
  func widthForColumnAtIndex(index:
  TSTableViewColumnIndex) -> CGFloat {
    return defaultTableViewColumnWidth
  }
  func xValueForColumnAtIndex(index:
  TSTableViewColumnIndex) -> CGFloat {
    var columnXValue: CGFloat = 0
    for previousIndex in 0 ..< index {
      let previousColumnWidth =
      widthForColumnAtIndex(previousIndex)
      columnXValue += previousColumnWidth
    }
    return columnXValue
  }
}
```

Listing 2: Grundlagen zur Erstellung von Reihen

```
class TSTableView: UIScrollView {
  ...
  private let defaultTableViewRowHeight: CGFloat =
  44
  private lazy var rowWidth: CGFloat = {
    return self.calculatedRowWidth()
  }()
  ...
  func heightForRowAtIndex(index:
  TSTableViewRowIndex) -> CGFloat {
    return defaultTableViewRowHeight
  }
  func yValueForRowAtIndex(index:
  TSTableViewRowIndex) -> CGFloat {
    var rowYValue = defaultTableViewHeaderHeight
    for previousIndex in 0 ..< index {
      let previousRowHeight =
      heightForRowAtIndex(previousIndex)
      rowYValue += previousRowHeight
    }
    return rowYValue
  }
  private func calculatedRowWidth() -> CGFloat {
    var rowWidth: CGFloat = 0
    for columnIndex in 0 ..< tableViewDelegate!.
    numberOfColumnsInTableView(self) {
      let columnWidth = widthForColumnAtIndex
      (columnIndex)
      rowWidth += columnWidth
    }
    return rowWidth
  }
}
```

Mit diesen grundlegenden Funktionen zur Kalkulation der korrekten x-Position und Breite einer Spalte der Table-View können wir nun mit ersten Inhalten der Tabelle beginnen. Beginnen wir dabei ganz oben; es sollen passende Header für jede Spalte der Tabelle erstellt werden. Diese Header werden in einer Reihe am oberen Rand der View angezeigt. Zur Umsetzung der Header werden Instanzen der Klasse *UILabel* verwendet, um auf diese Art und Weise Text als Überschrift für jede Spalte darzustellen. ▶

Wir werden die Label allerdings nicht direkt als Subviews der *TSTableView*-Instanz hinzufügen, sondern stattdessen die Referenzen auf diese Objekte einem eigenen entsprechenden Array hinzuzufügen. Das hat den Vorteil, dass zu einem späteren Zeitpunkt auf diese Labels wieder zugegriffen werden kann, indem einfach das entsprechende Array bemüht und abgefragt wird, ohne aufwendige Abfragen über die Subviews der Table-View zu durchlaufen.

Um die genannte Logik passend umzusetzen, implementieren wir zwei Methoden: Die eine erstellt ein Label für einen übergebenen Spaltenindex und fügt es dem beschriebenen Array hinzu. Die andere Methode durchläuft alle Spalten, über die die Table-View verfügt, und ruft dann die genannte Methode für jeden Spaltenindex einmal auf. Auf diese Art und Weise wird für jede Spalte ein Header-Label erstellt.

Listing 3: Erstellen von Reihen der Tabelle

```
class TSTableView: UIScrollView {
  ...
  private let
  defaultTableViewBackgroundColorForEvenRow =
  UIColor(red: 0.95, green: 0.95, blue: 0.95,
  alpha: 1)
  private let
  defaultTableViewBackgroundColorForUnevenRow =
  UIColor(red: 0.9, green: 0.9, blue: 0.9,
  alpha: 1)
  private var tableViewRowViews = [UIView]()
  ...
  private func createRows() {
    for index in 0 ..< tableViewDelegate!.
    numberOfRowsInTableView(self) {
      createRowViewAtIndex(index)
    }
  }
  private func createRowViewAtIndex(index:
  TSTableViewRowIndex) {
    let rowView = UIView(frame: CGRectMake(0,
    yValueForRowAtIndex(index), rowWidth,
    heightForRowAtIndex(index)))
    rowView.backgroundColor =
    backgroundColorForRowAtIndex(index)
    tableViewRowViews.append(rowView)
  }
  private func backgroundColorForRowAtIndex
  (index: TSTableViewRowIndex) -> UIColor {
    if index % 2 == 0 {
      return
      defaultTableViewBackgroundColorForEvenRow
    }
    return
    defaultTableViewBackgroundColorForUnevenRow
  }
}
```

Die Anzahl der Spalten wird über die Methode *numberOfColumnsInTableView:* des *tableViewDelegate*-Objekts ausgelesen. Zunächst werden die Labels für die Header ganz einfach formatiert und ein Standardtext gesetzt. Später optimieren wir diesen Code noch, um Titel für die Header auch aus einer entsprechenden Methode des Delegates auszulesen und für den Header zu setzen.

In Listing 1 sehen Sie eine mögliche Implementierung der genannten Funktionalität in der *TSTableView*-Klasse; der bisherige Code der Klasse ist aus Platzgründen in diesem Listing ausgeblendet. Wundern Sie sich auch nicht, dass die neu erstellten Methoden als *private* deklariert sind. Da diese rein für interne Zwecke der *TSTableView*-Klasse gedacht ist, sorgt die Deklaration als *private* dafür, dass sie nicht außerhalb der Klasse genutzt werden kann. Das Gleiche gilt für die Hilfs-Properties, die dazu dienen, bestimmte Informationen und Werte zu halten.

Eine Erklärung noch zur Kalkulation des Frames der Labels in der Methode *createHeaderLabelForColumnAtIndex::* Die x-Position wird anhand der Methode *xValueForColumnAtIndex:* ermittelt und dazu noch ein zusätzlicher Mindestabstand addiert. Der sorgt dafür, dass das Label nicht direkt am Rand klebt. Dieser Mindestabstand muss dann bei der Berechnung der Breite des Labels in doppelter Höhe wieder abgezogen werden; der erste Abzug ist notwendig, damit die Breite des Labels nicht über die eigentliche Breite der Spalte hinausragt. Der zweite sorgt dafür, dass das Label auch zum rechten Rand hin einen Mindestabstand besitzt. Der y-Wert entspricht für die Header-Label immer *0*, da diese immer am oberen Rand angezeigt werden sollen, und für die Höhe der Labels ist ebenfalls eine Property mit einem Standardwert definiert.

Reihen erstellen

Nachdem nun Funktionen zum Erstellen der Header unserer Table-View in einer ersten einfachen Variante zur Verfügung stehen, soll es nun mit den Reihen unserer Tabelle weitergehen. Zuoberst steht auch hier die essenzielle Frage nach der Größe und Position jeder Reihe. Bei zwei Informationen können wir es uns dabei einfach machen: der x-Position und der Breite. Die x-Position setzen wir immer auf *0*, da die Reihen immer am äußeren linken Rand beginnen sollen. Sie erstrecken sich dann über die komplette Breite der Tabelle. Diese Breite können wir ermitteln, indem wir alle Spalten der Tabelle durchlaufen, deren jeweilige Größe abfragen und diese dann nacheinander miteinander addieren.

Bei Letzterem gibt es eine Besonderheit zu beachten: Letzten Endes ist die Breite jeder Reihe gleich. Wir müssen diese Information also zunächst nur einmalig ermitteln und können sie dann auf alle Reihen anwenden. Daher wäre es aus Performance-Gründen unnötig, die genannte Berechnung für jede Reihe aufs Neue durchzuführen; das Ergebnis wäre stets dasselbe.

Aus diesem Grund erstelle ich eine Lazy Stored Property, die die Breite der Reihen speichert und zurückgibt. Beim erstmaligen Verwenden der Property ruft diese die passende Methode auf, die sich um die genannte Ermittlung der Reihenbreite kümmert. Anschließend hält diese Property den be-

rechneten Wert und wird ihn bei zukünftigen Aufrufen direkt zurückgeben, ohne eine erneute Berechnung durchführen zu müssen. Sollte der Wert der Property zu einem späteren Zeitpunkt geändert werden müssen (beispielsweise weil sich die Höhe unserer Reihen der Tabelle verändert hat), so kann ihr einfach wieder der Wert der Hilfsmethode zugewiesen werden.

Listing 2 zeigt die Ergänzung drei neuer Methoden, die wir somit ganz grundlegend für die Erstellung der Reihen unserer Tabellen benötigen: *heightForRowAtIndex:* liefert die Höhe einer Reihe zurück. Äquivalent zur Breite der Spalten geben wir hier zunächst einmal einen in der Klasse festgelegten Standardwert zurück. *yValueForRowAtIndex:* liefert uns die Position der y-Achse der Reihe zurück, deren Index als Parameter abgefragt wird. Dabei gehen wir ähnlich vor wie bei der x-Position der Spalten: Wir durchlaufen alle vorherigen Reihen, ermitteln deren jeweilige Größe und addieren diese zusammen. Zu beachten ist dabei, dass der Startwert für die y-Position der ersten Reihe nicht *0*, sondern die Höhe des Headers der Tabelle ist; schließlich soll die erste Reihe unterhalb des Headers und nicht auf diesem platziert werden. *calculatedRowWidth* zu guter Letzt kalkuliert die Breite für jede einzelne Reihe. Das daraus resultierende Ergebnis wird auch innerhalb einer Lazy Stored Property namens *rowWidth* verwendet, die für das Auslesen der Breite verwendet wird.

Nun geht es noch um das Erstellen der eigentlichen Reihen-Objekte. Ich verwende dazu einfache *UIView*-Instanzen, eine für jede Reihe. Um sie voneinander zu unterscheiden, wird ihnen abwechselnd eine andere Hintergrundfarbe zugewiesen; diese wird mittels einer Hilfsmethode ermittelt und zurückgegeben.

Das Erstellen der Reihen ist sehr ähnlich wie das der Header für die Spalten. Eine Methode durchläuft in einer *for*-Schleife die Anzahl der Reihen, die über den Delegate ermittelt werden. Für jede dieser Reihen wird dann eine Methode aufgerufen, um ein passendes UIView-Objekt zu erzeugen. Der Frame für jede Reihe wird mittels der zuvor erstellten Methoden korrekt ermittelt. Am Ende werden auch die Reihen-Objekte einem neuen Array namens *tableViewRowViews* hinzugefügt, um diese darüber zu halten und einen Zugriffspunkt zu besitzen, falls es nötig sein sollte, eine einzelne Reihe später noch einmal zu verändern. Listing 3 zeigt die neuen Properties und Methoden, die zur Umsetzung der genannten Aufgabe benötigt werden.

Zellen erstellen

Um unsere eigene Table-View-Klasse nun zu guter Letzt in einer ersten Version abzuschließen, muss noch die passende Logik bereitgestellt werden, um einzelne Zellen innerhalb der Tabelle mit Inhalten zu füllen. Hierbei handelt es sich um die eigentlich wichtigsten Elemente der Tabelle, enthalten diese doch die anzuzeigenden Inhalte.

Um darum möglichst flexibel in der Darstellung und Verwendung der *TSTableView*-Klasse zu bleiben, verwenden wir als Basisklasse für jede Zelle ein UIView-Objekt. Das erlaubt es uns, jede beliebige Art von View als Zelle zu verwenden. Wie bei den anderen Elementen halten wir die zugrunde ▶

developer media

developer media Trainings

Neue Trainings für Developer

Node.js, io.js & Co. – Entwickeln für die Cloud
Trainer: Golo Roden
3 Tage, Termin & Ort n. V.

Webentwicklung mit ASP.NET MVC & Web API
Trainer: David Tielke
3 Tage, 13.-15.07.2016, Köln

PHP-Security
Trainer: Arne Blankerts
2 Tage, Hamburg,
Termin nach Vereinbarung

Domain Driven Design mit PHP
Trainer: Stefan Priebsch
2 Tage, München, Termin n. V.

Webanwendungen mit HTML, CSS3 und JavaScript
Trainer: David Tielke
2 Tage, 11.-12.07.2016, Köln

PHPUnit erfolgreich einsetzen
Trainer: Sebastian Bergmann
2 Tage, Köln, Termin n. V.

Agile Produktentwicklung
Trainer: Björn Schotte
2 Tage, Termin & Ort
nach Vereinbarung

Weitere Informationen und Anmeldung unter
www.developer-media.de/trainings

liegende Logik erst einmal relativ einfach und geben einen Standardtext in Form eines Labels in jeder Zelle aus. Damit ist die grundlegende Funktionsweise der Table-View sichergestellt, ohne sich direkt mit dem korrekten Datenhandling des Delegates auseinandersetzen zu müssen.

Um den genannten Standardtext pro Zelle auszugeben – der einfach die Spalte und die Reihe der jeweiligen Zelle umfasst – verwenden wir Instanzen der Klasse *UILabel* und packen diese zusätzlich noch in eine separate UIView. Die UIView benutzen wir dabei als Rahmen, in den wir das Label so einfügen, das es gut in die Zelle passt. Schließlich wird die UIView die gesamte Breite und Höhe der Zelle ausfüllen, und damit unser Text in Form des UILabels nicht an den Ecken klebt, passen wir dessen x-Wert und Breite entsprechend an, damit es etwas vom Rand der umschließenden UIView absteht (ähnlich wie bei den Spalten-Headern).

Um die Zellen zu erstellen, durchlaufen wir alle Reihen der Tabelle in einer *for*-Schleife und darin noch einmal alle Spalten, ebenfalls innerhalb einer *for*-Schleife. Daraus können wir eine Instanz unserer zu Beginn erstellten Structure *TSTableViewCellCoordinate* erstellen, der wir pro Zelle die passende Reihe und Spalte zuweisen. Aus dieser Information heraus können wir dann in einer separaten Methode die dazu passende Zelle erstellen.

Wie auch schon bei den Headern und den Zeilen packen wir alle so erstellten Zellen-UIView-Objekte in ein Array, das die Referenzen auf jedes Zellen-Objekt hält und das dazu genutzt werden kann, um später auf die Views einzelner Zellen zuzugreifen. Auch werden wir dieses sowie die anderen Arrays dann dazu nutzen, die einzelnen UIView-Objekte der Table-View hinzuzufügen.

Alle Ansichten der Table-View hinzufügen

Damit ist nun all der grundlegende Code erstellt, um die verschiedenen Teile der Tabelle – Header, Reihen, Spalten und Zellen – zu erstellen. Was nun noch fehlt, ist, all das zusammenzuführen und dafür zu sorgen, dass all die Views, die wir in verschiedenen Arrays zwischengelagert haben, nun auch der Table-View als Subviews hinzugefügt werden. Zu diesem Zweck implementieren wir am besten verschiedene Methoden, die jeweils die Views aus einem spezifischen Array – zum Beispiel das der Header oder der Reihen – durchlaufen und als Subviews der Table-View hinzufügen. Diese Methoden werden dann von einer umfassenden Methode aufgerufen, die somit die komplette Aktualisierung und Darstellung der Table-View übernimmt.

Bei dieser umfassenden Methode ist zu beachten, dass diese von außen aufrufbar sein sollte, falls der Nutzer selbst ein Update der Table-View-Inhalte anstoßen möchte. Dann muss die Methode aber zuvor die existierenden Subviews der Table-View entfernen, da ansonsten die aktualisierten Subviews über die bestehenden gelegt werden.

Bleibt noch die Frage, wann die Aktualisierung der Table-View idealerweise angestoßen werden soll. Wie bereits geschrieben ergibt es definitiv Sinn, die dafür neu zu erstellende Methode öffentlich zu machen, damit sie auch von außen genutzt werden kann. Doch natürlich ist es nicht die schönste Lösung, den Nutzer nach Erstellen einer neuen Table-View und dem Zuweisen eines Delegates selbst auch noch die Methode zum Aktualisieren der Inhalte aufrufen zu lassen.

Aus diesem Grund überarbeite ich an dieser Stelle auch noch einmal unsere anfangs erstellte *tableViewDelegate*-Property und setze einen Property Observer, der immer dann gefeuert wird, wenn der Delegate der *TSTableView*-Instanz geändert wird. Dann nämlich ist es im Prinzip immer notwendig, auch die Table-View selbst komplett neu aufzubauen. Weisen wir also einem *TSTableView*-Objekt einen Delegate zu, wird auch automatisch die Tabelle mit den korrekten Inhalten neu aufgebaut. In Listing 4 finden Sie die Implementierung der genannten Ergänzungen und Änderungen.

Ein erster Test

Damit ist bereits einiges an Code zusammengekommen; Zeit, einmal den aktuellen Stand konkret zu prüfen und zu testen. Aus diesem Grund erstellen wir nun in einem nächsten Schritt

Listing 4: Hinzufügen der Ansichten zur Table-View

```
class TSTableView: UIScrollView {
  var tableViewDelegate: TSTableViewDelegate? {
    didSet {
      updateTableView()
    }
  }
  ...
  func updateTableView() {
    for view in subviews {
      view.removeFromSuperview()
    }
    addTableViewHeaderViews()
    addTableViewRowViews()
    addTableViewCellViews()
  }
  private func addTableViewHeaderViews() {
    createHeaderViews()
    for headerLabel in tableViewHeaderLabels {
      addSubview(headerLabel)
    }
  }
  private func addTableViewRowViews() {
    createRows()
    for rowView in tableViewRowViews {
      addSubview(rowView)
    }
  }
  private func addTableViewCellViews() {
    createCells()
    for cellView in tableViewCellViews {
      addSubview(cellView)
    }
  }
}
```

einmal eine Klasse, die konform zu unserem *TSTableViewDelegate*-Protokoll ist und somit die Methoden mit der Anzahl der Spalten und Reihen implementieren muss. Anschließend erstellen wir eine Instanz der *TSTableView*-Klasse mit einem festen Frame und weisen ihr ein Objekt unserer neuen, protokollkonformen Klasse zu.

Das führt dazu, dass die Table-View den implementierten Code ausführt und sich selbst vollständig aufbaut. Wir erhalten daraufhin eine vollständige Tabellenansicht in genau der Form, wie wir sie bis zu diesem Zeitpunkt umgesetzt haben. Das folgende Listing zeigt den dazu von mir verwendeten Testcode, der eine neue, voll funktionsfähige Instanz unserer neuen *TSTableView*-Klasse erstellt und darstellt:

```
class MyTableViewDelegate: TSTableViewDelegate {
  func numberOfColumnsInTableView(tableView:
  TSTableView) -> Int {
    return 3
  }
  func numberOfRowsInTableView(tableView: TSTableView)
  -> Int {
    return 3
  }
}
let myTableViewDelegate = MyTableViewDelegate()
let tableView = TSTableView(frame:
CGRectMake(0, 0, 600, 200))
tableView.tableViewDelegate = myTableViewDelegate
```

Mit unserem bisherigen Code ist es nun also schon einmal möglich, dynamisch eine neue Tabelle zu erstellen und dabei festzulegen, über wie viele Spalten und Reihen und damit auch Zellen diese verfügt. Das ist gut und schön, doch fehlt uns an dieser Stelle noch jeglicher dynamischer Content, den unsere Tabelle anzeigen kann.

Protokoll erweitern

Ein guter Zeitpunkt also, um unser Protokoll um neue Methoden zu erweitern und unsere *TSTableView*-Klasse entsprechend anzupassen, damit wir eigene Titel für die Header und die Zellen festlegen können. Damit ein Delegate diese Informationen liefern kann, benötigt er zwei Informationen:

- Um welche TSTableView-Instanz handelt es sich?
- Um welche Spalte beziehungsweise Zelle handelt es sich?

Letztere Information wird bei Spalten in Form des *TSTableViewColumnIndex* definiert, bei Zellen in Form der *TSTableViewCellCoordinate*-Structure. Diese Informationen müssen also über die zu deklarierenden Methoden im *TSTableViewDelegate*-Protokoll übergeben werden, damit der Delegate weiß, an welcher Stelle eine Information angefragt wird und zurückgegeben werden soll.

Das folgende Listing zeigt eine mögliche Deklaration der beiden benötigten Methoden innerhalb des *TSTableViewDelegate*-Protokolls. Als Rückgabewert erwarten die Methoden einen String, der dann als Titel für die Header beziehungsweise den Inhalt der Zellen verwendet werden soll:

Zahlen		Buchstaben
1	-	A
2	-	B
3	-	C

Ändern der Table-View-Inhalte durch den Delegate (Bild 3)

```
protocol TSTableViewDelegate {
  func numberOfRowsInTableView(tableView: TSTableView)
  -> Int
  func numberOfColumnsInTableView(tableView:
  TSTableView) -> Int
  func tableView(tableView: TSTableView,
  titleForColumnHeaderAtIndex index:
  TSTableViewColumnIndex) -> String
  func tableView(tableView: TSTableView,
  titleForCellWithCoordinates cellCoordinates:
  TSTableViewCellCoordinate) -> String
}
```

Bevor wir die neu hinzugefügten Methoden in unserer Beispiel-Delegate-Klasse *MyTableViewDelegate* implementieren, beschäftigen wir uns zunächst einmal damit, wie wir diese neue Methoden innerhalb unserer *TSTableView*-Klasse korrekt einsetzen und aufrufen, um die Titel der Header und Zellen entsprechend zu aktualisieren. Beginnen wir dabei mit den Headern.

Bisher wird den Headern der Tabelle in der Methode *createHeaderLabelForColumnAtIndex*: ein statischer Text zugewiesen. Das ändern wir nun, indem wir stattdessen der Property *text* des *headerLabel*-Objekts innerhalb der genannten Methode den Rückgabewert der neu im *TSTableViewDelegate*-Protokoll deklarierten Methode *tableView:titleForColumnHeaderAtIndex:* zuweisen.

Das folgende Listing zeigt die entsprechend aktualisierte Methode *createHeaderLabelForColumnAtIndex:* der *TSTableView*-Klasse:

```
private func createHeaderLabelForColumnAtIndex(index:
TSTableViewColumnIndex) {
  let headerLabel = UILabel(frame: CGRectMake
  (xValueForColumnAtIndex(index) +
  defaultTableViewCellIndention, 0,
  widthForColumnAtIndex(index) -
  defaultTableViewCellIndention * 2,
  defaultTableViewHeaderHeight))
  headerLabel.font = UIFont.boldSystemFontOfSize(17)
  headerLabel.text = tableViewDelegate!.tableView(self,
  titleForColumnHeaderAtIndex: index)
  tableViewHeaderLabels.append(headerLabel)
}
```

Äquivalent dazu verfahren wir in der Methode *createCellViewForCellWithCoordinates*:, in der bisher der statische Titel ▶

für eine Zelle gesetzt wird. Hier verwenden wir stattdessen den zurückgegebenen String der Delegate-Methode *tableView:titleForCellWithCoordinates:* und weisen diesen dem Label einer Zelle als Titel zu. Das nachfolgende Listing zeigt die aktualisierte Implementierung der Methode *createCellViewForCellWithCoordinates:*

```
private func createCellViewForCellWithCoordinates
(cellCoordinates: TSTableViewCellCoordinate) {
  let cellView = UIView(frame: CGRectMake
  (xValueForColumnAtIndex(cellCoordinates.columnIndex),
  yValueForRowAtIndex(cellCoordinates.rowIndex),
  widthForColumnAtIndex(cellCoordinates.columnIndex),
  heightForRowAtIndex(cellCoordinates.rowIndex)))
  let cellLabel = UILabel(frame: CGRectMake
  (defaultTableViewCellIndention, 0,
  widthForColumnAtIndex(cellCoordinates.columnIndex) -
  defaultTableViewCellIndention * 2, heightForRowAtIndex
  (cellCoordinates.rowIndex)))
  cellLabel.text = tableViewDelegate!.tableView(self,
  titleForCellWithCoordinates: cellCoordinates)
  cellView.addSubview(cellLabel)
  tableViewCellViews.append(cellView)
}
```

Listing 5: TSTableViewDelegate-Aktualisierung

```
@objc protocol TSTableViewDelegate {
  func numberOfRowsInTableView(tableView:
  TSTableView) -> Int
  func numberOfColumnsInTableView(tableView:
  TSTableView) -> Int
  func tableView(tableView: TSTableView,
  titleForColumnHeaderAtIndex index:
  TSTableViewColumnIndex) -> String
  func tableView(tableView: TSTableView,
  titleForCellWithCoordinates cellCoordinates:
  TSTableViewCellCoordinate) -> String
  optional func tableView(tableView: TSTableView,
  heightForRowAtIndex index: TSTableViewRowIndex)
  -> CGFloat
  optional func tableView(tableView: TSTableView,
  widthForColumnAtIndex index:
  TSTableViewColumnIndex) -> CGFloat
}
```

Listing 6: Notwendige Anpassungen

```
class TSTableViewCellCoordinate: NSObject {
  var columnIndex: TSTableViewColumnIndex
  var rowIndex: TSTableViewRowIndex
  init(columnIndex: TSTableViewColumnIndex,
  rowIndex: TSTableViewRowIndex) {
    self.columnIndex = columnIndex
    self.rowIndex = rowIndex
  }
}
...
class MyTableViewDelegate: NSObject,
TSTableViewDelegate {
  ...
}
```

Um die aus diesen kleinen Änderungen resultierenden mächtigen Möglichkeiten einmal zu demonstrieren, habe ich die Implementierung der Delegate-Klasse *MyTableViewDelegate* entsprechend überarbeitet. Die Anzahl an Reihen und Spalten bleibt dieselbe. Für die erste und letzte Spalte wird nun ein eigener Titel verwendet, während der Titel der mittleren Spalte leer bleibt. In der ersten Spalte werden dann pro Reihe nacheinander Zahlen ausgegeben, in der letzten Spalte nacheinander Buchstaben. In der mittleren Spalte wird einfach für jede Zeile ein Minuszeichen angezeigt.

Die größte Änderung wurde somit innerhalb unserer eigenen Delegate-Klasse vorgenommen, während sich an der *TSTableView* selbst nur eine Kleinigkeit geändert hat. Dennoch sind die Auswirkungen auf den Aufbau und das Aussehen der so erzeugten *TSTableView*-Instanz enorm (Bild 3).

Genau das ist ja aber auch der Sinn der Sache: Der Delegate soll als primäre Anlaufstelle für den Aufbau und das Aussehen einer beliebigen TSTableView-Instanz fungieren, und diesen Grundstein haben wir so weit in bestem Maße umgesetzt.

Definieren optionaler Delegate-Methoden

Im nächsten Schritt wollen wir es nun ermöglichen, für jede Reihe und jede Spalte eine eigene Höhe beziehungsweise Breite festzulegen. Bisher verwenden wir dafür einen eigens definierten Standardwert, der nun um eine optionale Delegate-Methode ergänzt werden soll. Das bedeutet, dass diese Delegate-Methode zum Setzen der Reihenhöhe und der Spaltenbreite nicht zwingend von einem Delegate implementiert werden muss (in diesem Fall bleibt alles beim Alten und die Table-View greift stattdessen auf ihren Standardwert für die Größe zurück), aber auf Wunsch implementiert werden kann, um selbst die Größe der Table-View-Elemente zu definieren.

Derart optionale Methoden in Protokollen werden in Swift mit Hilfe des *optional*-Keywords umgesetzt. Befindet sich dieses Schlüsselwort vor der Deklaration einer Methode in einem Protokoll, so ist diese Methode optional und muss von einem Delegate, der konform zu eben jenem Protokoll ist, nicht implementiert werden.

Dabei gibt es aber einen kleinen Haken: Bisher erlaubt Swift optionale Methoden in Protokollen nur dann, wenn das Protokoll mit Hilfe des Schlüsselworts *@objc* deklariert ist. Dieses Schlüsselwort wird typischerweise bei Klassen verwendet, die auch aus Objective-C-Code heraus aufgerufen und angesprochen werden können.

Prinzipiell ist dieser Umstand nicht weiter tragisch, doch er zwingt uns dazu, weitere Änderungen an unserem bestehen-

den Code vorzunehmen. Nehmen wir dazu beispielsweise unsere Structure *TSTableViewCellCoordinate:* Diese wird in der Methode *tableView:titleForCellWithCoordinates:* unseres *TSTableViewDelegate*-Protokolls verwendet. Objective-C kann mit derartigen Structures nichts anfangen, und das wird nun zu einem Problem, da wir eben jenes Protokoll mit *@objc* deklariert haben. Das zwingt uns dazu, dass alle Inhalte dieses Protokolls in jedem Fall von Objective-C-Code verstanden werden müssen. Es bleibt uns also nichts anderes übrig, als unsere bisherige *TSTableViewCellCoordinate*-Structure in eine Klasse umzuwandeln (und in diesem Zuge noch einen notwendigen Initializer zu ergänzen). Auch muss *TSTableViewCellCoordinate* nun zwingend von *NSObject* erben.

Entsprechendes gilt nun übrigens auch für unsere Delegate-Klasse *MyTableViewDelegate*. Diese kann nun nur noch konform zum *TSTableViewDelegate*-Protokoll sein, wenn sie ebenfalls von *NSObject* erbt.

Listing 5 zeigt die Aktualisierung des TSTableViewDelegate-Protokolls, inklusive der zwei neuen optionalen Methoden zum Abfragen der Spaltenbreite und Reihenhöhe. Listing 6 zeigt die notwendige Umwandlung der *TSTableViewCellCoordinate*-Structure in eine Klasse und die Aktualisierung der Klassendefinition von *MyTableViewDelegate*.

Spaltenbreiten und Reihenhöhen

Mit dieser grundlegenden Vorbereitung können wir nun die Aktualisierung der *TSTableView*-Klasse in Angriff nehmen, um den neuen optionalen Protokoll-Methoden Rechnung zu tragen. Betrachten wir dabei zunächst die Spaltenbreite, die im Protokoll über die optionale Methode *tableView:widthForColumnAtIndex:* definiert ist.

Header 0	Header 1	Header 2
Reihe 0 | Spalte 0	Reihe 0 | Spalte 1	Reihe 0 | Spalte 2
Reihe 1 | Spalte 0	Reihe 1 | Spalte 1	Reihe 1 | Spalte 2
Reihe 2 | Spalte 0	Reihe 2 | Spalte 1	Reihe 2 | Spalte 2

Table-View mit angepassten Breiten und Reihenhöhen (Bild 4)

Wir besitzen innerhalb unserer *TSTableView*-Klasse bereits eine Methode namens *widthForColumnAtIndex:*, die wir dazu verwenden, für jede Spalte die korrekte Breite zurückzugeben. Das ist also die optimale Stelle, unsere bisherige Logik für die neue Protokollmethode zu aktualisieren.

Dabei zu beachten ist, dass es sich bei der aufzurufenden Methode um eine optionale Methode handelt. Das bedeutet, wir können diese nicht – so, wie es bisher bei den anderen Methoden des *TSTableViewDelegate*-Protokolls der Fall war – einfach auf unserem *tableViewDelegate*-Objekt aufrufen. Schließlich wissen wir nicht, ob das zugewiesene Delegate-Objekt diese Methode auch tatsächlich implementiert hat.

Um diese Frage zu klären, verwenden wir die Optional-Chaining-Syntax von Swift. Innerhalb eines *if let*-Blocks prüfen wir, ob die gewünschte Methode tatsächlich existiert, aufgerufen werden kann und entsprechend einen Wert zurückliefert. Ist das der Fall, gibt die Methode *widthForColumnAtIndex*: ebenso genau jenen Wert zurück, den wir über den Delegate erhalten haben. Andernfalls wird wie gewohnt der Standardwert der *TSTableView*-Klasse als Rückgabewert der Methode verwendet. ▶

Listing 7: Anpassung der Spaltenbreiten und Reihenhöhen

```
class MyTableViewDelegate: NSObject,
TSTableViewDelegate {
  func numberOfColumnsInTableView(tableView:
  TSTableView) -> Int {
    return 3
  }
  func numberOfRowsInTableView(tableView:
  TSTableView) -> Int {
    return 3
  }
  func tableView(tableView: TSTableView,
  titleForColumnHeaderAtIndex index:
  TSTableViewColumnIndex) -> String {
    return "Header \(index)"
  }
  func tableView(tableView: TSTableView,
  titleForCellWithCoordinates cellCoordinates:
  TSTableViewCellCoordinate) -> String {
    return "Reihe \(cellCoordinates.rowIndex) |
    Spalte \(cellCoordinates.columnIndex)"
  }
  func tableView(tableView: TSTableView,
  widthForColumnAtIndex index:
  TSTableViewColumnIndex) -> CGFloat {
    if index == 0 {
      return 200
    }
    return 300
  }
  func tableView(tableView: TSTableView,
  heightForRowAtIndex index: TSTableViewRowIndex) ->
  CGFloat {
    if index == 1 {
      return 88
    }
    return 44
  }
}
let myTableViewDelegate = MyTableViewDelegate()
let tableView = TSTableView(frame: CGRectMake(0, 0,
800, 220))
tableView.tableViewDelegate = myTableViewDelegate
```

Das nachfolgende Listing zeigt die entsprechende Überarbeitung der Methode *widthForColumnAtIndex:*

```
func widthForColumnAtIndex(index:
TSTableViewColumnIndex) -> CGFloat {
  if let columnWidth = tableViewDelegate!.
  tableView?(self, widthForColumnAtIndex: index) {
    return columnWidth }
  return defaultTableViewColumnWidth
}
```

Bei der Höhe der einzelnen Reihen haben wir bereits mit *heightForRowAtIndex:* eine passende Methode vorbereitet, die die Höhe jeder einzelnen Reihe ermittelt und zurückgibt. Wir können diese Methode nun nutzen, um zu prüfen, ob die optionale Methode *tableView:heightForRowAtIndex:* über unser *tableViewDelegate*-Objekt zur Verfügung steht. Falls ja, rufen wir sie auf und liefern das Ergebnis als Rückgabewert der Methode zurück. Andernfalls greifen wir stattdessen wie bisher auf unseren Standardwert zu:

```
func heightForRowAtIndex(index: TSTableViewRowIndex) ->
CGFloat {
  if let heightForRowAtIndex = tableViewDelegate!.
  tableView?(self, heightForRowAtIndex: index) {
    return heightForRowAtIndex }
  return defaultTableViewRowHeight
}
```

Implementieren wir nun die beiden Methoden oder eine davon in unserer *MyTableViewDelegate*-Klasse, können wir

Listing 8: Erweitern des TSTableViewDelegate

```
@objc protocol TSTableViewDelegate {
  func numberOfRowsInTableView(tableView:
  TSTableView) -> Int
  func numberOfColumnsInTableView(tableView:
  TSTableView) -> Int
  func tableView(tableView: TSTableView,
  titleForColumnHeaderAtIndex index:
  TSTableViewColumnIndex) -> String
  func tableView(tableView: TSTableView,
  titleForCellWithCoordinates cellCoordinates:
  TSTableViewCellCoordinate) -> String
  optional func tableView(tableView: TSTableView,
  heightForRowAtIndex index: TSTableViewRowIndex)
  -> CGFloat
  optional func tableView(tableView: TSTableView,
  widthForColumnAtIndex index:
  TSTableViewColumnIndex) -> CGFloat
  optional func tableView(tableView: TSTableView,
  viewForCellWithCoordinates cellCoordinates:
  TSTableViewCellCoordinate) -> UIView?
}
```

Listing 9: Auslesen der optionalen Zellen-Ansichten

```
private func createCellViewForCellWithCoordinates
(cellCoordinates: TSTableViewCellCoordinate) {
  if let cellView = tableViewDelegate!.
  tableView?(self, viewForCellWithCoordinates:
  cellCoordinates) {
    tableViewCellViews.append(cellView)
    return
  }
  createLabelForCellWithCoordinates
  (cellCoordinates)
}
private func createLabelForCellWithCoordinates
(cellCoordinates: TSTableViewCellCoordinate) {
  let cellView = UIView(frame: CGRectMake
  (xValueForColumnAtIndex(cellCoordinates.
  columnIndex), yValueForRowAtIndex(cellCoordinates.
  rowIndex), widthForColumnAtIndex(cellCoordinates.
  columnIndex), heightForRowAtIndex(cellCoordinates.
  rowIndex)))
let cellLabel = UILabel(frame: CGRectMake
(defaultTableViewCellIndention, 0,
widthForColumnAtIndex(cellCoordinates.columnIndex)
- defaultTableViewCellIndention * 2,
heightForRowAtIndex(cellCoordinates.rowIndex)))
cellLabel.text = tableViewDelegate!.tableView(self,
titleForCellWithCoordinates: cellCoordinates)
cellView.addSubview(cellLabel)
tableViewCellViews.append(cellView)
}
```

feststellen, dass sich das Aussehen der Table-View entsprechend verändert. Listing 7 zeigt dazu ein Beispiel. Die erste Spalte ist schmaler als die letzten beiden, und die mittlere Reihe ist am höchsten (Bild 4).

Verwenden von eigenen Views für Zellen

Zum Abschluss der Konfiguration unserer eigenen Table-View-Klasse soll noch eine letzte Logik implementiert werden: Der Nutzer soll die Möglichkeit haben, für einzelne Zellen eine eigene View für die Anzeige zu verwenden (anstelle des bisher verwendeten einfachen Textes).

Dazu braucht es zunächst einmal eine neue Methode innerhalb unseres TSTableViewDelegate-Protokolls, die eine entsprechende Funktion zur Verfügung stellt. Diese übergibt die Koordinaten der jeweiligen Zelle als Parameter und erwartet ein optionales UIView-Objekt zurück.

Warum optional? Nun ja, womöglich möchten Sie nur für einzelne Zellen der Tabellen eine eigene UIView verwenden und für den Rest auf die bereits bekannte Textansicht setzen. Wäre der Rückgabewert für eine eigene Zellen-View dann nicht optional, müssten Sie dafür Sorge tragen, bei allen Zellen eine eigene Ansicht zurückzuliefern; selbst bei denen, bei denen Sie lediglich den ganz normalen Text anzeigen möch-

Header 0	Header 1	Header 2
Reihe 0 | Spalte 0	Reihe 0 | Spalte 1	Reihe 0 | Spalte 2
Reihe 1 | Spalte 0		Reihe 1 | Spalte 2
Reihe 2 | Spalte 0	Reihe 2 | Spalte 1	Reihe 2 | Spalte 2

Darstellung der eigens erstellten View für eine Zelle (Bild 5)

ten. In diesem Fall würden Sie also sowohl die Delegate-Methode zum Zurückliefern einer eigenen Zellenansicht als auch die Delegate-Methode zum Zurückliefern eines Titels für eine bestimmte Zelle implementieren und je nach Koordinaten der Zelle die passende Information zurückgeben.

Listing 8 zeigt, wie die Deklaration einer solchen Methode innerhalb des *TSTableViewDelegate*-Protokolls aussehen kann. Die Methode ist als optional deklariert und muss somit vom Delegate-Objekt nicht implementiert werden. Mit Hilfe der übergebenen *TSTableView*- und *TSTableViewCellCoordinate*-Instanzen kann das Delegate-Objekt dann eindeutig die zugrunde liegende Tabelle mitsamt Zelle identifizieren und darauf basierend die korrekte Information zurückgeben. Als Nächstes geht es darum, den bisherigen Code zur Erstellung der Zellen-Views der Klasse *TSTableView* entsprechend zu aktualisieren. Dazu lagert man zunächst den Code, der das bisher verwendete Label erstellt und konfiguriert hat, in eine eigene Methode aus. Innerhalb der bisherigen Methode *createCellViewForCellWithCoordinates:* prüft man dann, ob das *tableViewDelegate*-Objekt einen validen Wert für die Methode *tableView:viewForCellWithCoordinates:* zurückliefert. Ist das der Fall, wird genau diese View dem Array *tableViewCellViews*, das alle Zellen-View-Objekte enthält, hinzugefügt und die Methode verlassen. Andernfalls wird die neu erstellte Methode zum Erstellen des Standard-Labels aufgerufen, die die bereits bekannten Zellen-Ansichten erstellt und ihrerseits dem *tableViewCellViews*-Array hinzugefügt. Die entsprechende Aktualisierung des Codes der Klasse *TSTableView* sehen Sie in Listing 9.

Überarbeitung der Klasse

Damit ist die *TSTableView* für die Verwendung von View-Objekten für einzelne Zellen vorbereitet. Listing 10 zeigt eine dazu passende Überarbeitung unserer *MyTableViewDelegate*-Klasse, die nun eben jene Methode implementiert und lediglich für die Zelle der zweiten Spalte in der zweiten Reihe eine einfache UIView mit blauem Hintergrund zurückliefert; für alle restlichen Zellen wird weiterhin der standardmä- ▶

Listing 10: Erstellen einer eigenen View für eine Zelle

```
class MyTableViewDelegate: NSObject,
TSTableViewDelegate {
  func numberOfColumnsInTableView(tableView:
  TSTableView) -> Int {
    return 3
  }
  func numberOfRowsInTableView(tableView:
  TSTableView) -> Int {
    return 3
  }
  func tableView(tableView: TSTableView,
  titleForColumnHeaderAtIndex index:
  TSTableViewColumnIndex) -> String {
    return "Header \(index)"
  }
  func tableView(tableView: TSTableView,
  titleForCellWithCoordinates cellCoordinates:
  TSTableViewCellCoordinate) -> String {
    return "Reihe \(cellCoordinates.rowIndex) |
    Spalte \(cellCoordinates.columnIndex)"
  }
  func tableView(tableView: TSTableView,
  widthForColumnAtIndex index:
  TSTableViewColumnIndex) -> CGFloat {
    if index == 0 {
      return 200
    }
    return 300 }
  func tableView(tableView: TSTableView,
  heightForRowAtIndex index: TSTableViewRowIndex) ->
  CGFloat {
    if index == 1 {
      return 88
    }
    return 44
  }
  func tableView(tableView: TSTableView,
  viewForCellWithCoordinates cellCoordinates:
  TSTableViewCellCoordinate) -> UIView? {
    if cellCoordinates.columnIndex == 1 &&
    cellCoordinates.rowIndex == 1 {
     let myViewFrame = CGRectMake(tableView.
     xValueForColumnAtIndex(cellCoordinates.
     columnIndex), tableView.yValueForRowAtIndex
     (cellCoordinates.rowIndex),
     tableView.widthForColumnAtIndex
     (cellCoordinates.columnIndex), tableView.
     heightForRowAtIndex(cellCoordinates.
     rowIndex))
     let myView = UIView(frame: myViewFrame)
     myView.backgroundColor = UIColor.blueColor()
      return myView
    }
    return nil
  }
}
```

ßige Text für die Anzeige verwendet. Beachtenswert bei dieser Implementierung ist dabei die Kalkulation des Frames der eigenen UIView, die für die Zelle verwendet wird. Position und Größe können ohne eigene Berechnungen direkt anhand der bereits von uns erstellten Methoden der TSTableView zur Ermittlung der x- und y-Positionen sowie der Höhe und der Breite ermittelt werden. Bild 5 zeigt das Ergebnis der überarbeiteten Delegate-Implementierung.

Auslesen der Table-View-Subviews

Zum Abschluss dieses Artikels wollen wir noch eine letzte Optimierung vornehmen, um die Flexibilität im Umgang mit unserer eigens erstellten Table-View zu erhöhen. Dazu stellen wir noch drei zusätzliche öffentliche Methoden bereit, über die man auf die einzelnen Views der Tabelle zugreifen kann – schließlich haben wir die Ansichten für die Header, Reihen und Zellen extra aus diesem Grund zu Beginn in entsprechenden Arrays gespeichert.

Die Methoden sind dabei allesamt einfach gehalten. Sie erhalten einen Index für die Reihe oder Spalte beziehungsweise Koordinaten für die Zelle, deren View-Objekt zurückgeliefert werden soll, und prüfen dann innerhalb des zugehörigen Arrays, ob es ein View-Objekt für den passenden Index gibt. Wenn ja, wird dieser zurückgegeben, andernfalls *nil*. Somit kommt es nicht zu einem Absturz, falls über eine solche Methode ein Index abgefragt wird, der nicht in den entsprechenden Arrays existiert.

Etwas knifflig gestaltet sich dabei die Ermittlung des Index für eine Zelle. Während man bei Spalten und Reihen direkt den übergebenen Parameter als Index zum Zugriff auf das entsprechende Array verwenden kann, geht das bei Zellen nicht so ohne Weiteres, da sich diese ja aus einem Spalten- und einem Reihenindex zusammensetzen.

Es muss also zuvor der korrekte Index für das Array *tableViewCellViews* kalkuliert werden. Das ist aber erfreulicherweise nicht allzu schwer: Man multipliziert den Reihenindex der übergebenen Koordinaten einfach mit der Anzahl der Spalten der Tabelle und addiert den übergebenen Spaltenindex dazu. Das entspricht genau dem Index in der Reihenfolge, wie die Zellen-Views auch bei der Erstellung dem *tableViewCellViews*-Array hinzugefügt werden. Listing 11 zeigt die Implementierung der entsprechenden Methoden.

Listing 11: Erweitern des TSTableViewDelegate

```
class TSTableView: UIScrollView {
  ...
  func headerLabelForColumnAtIndex(index:
  TSTableViewColumnIndex) -> UILabel? {
    if tableViewHeaderLabels.indices.
    contains(index) {
      return tableViewHeaderLabels[index]
    }
    return nil
  }
  func rowViewAtIndex(index: TSTableViewRowIndex)
  -> UIView? {
    if tableViewRowViews.indices.contains(index) {
      return tableViewRowViews[index]
    }
    return nil }
  func cellViewForCellWithCoordinates
  (cellCoordinates: TSTableViewCellCoordinate) ->
  UIView? {
    let tableViewCellViewsIndex = cellCoordinates.
    rowIndex * tableViewDelegate!.
    numberOfColumnsInTableView(self) +
    cellCoordinates.columnIndex
    if tableViewCellViews.indices.contains
    (tableViewCellViewsIndex) {
      return tableViewCellViews
      [tableViewCellViewsIndex] }
    return nil
  }
}
```

Links zum Thema

- TSTableViewKit-Framework auf GitHub
  https://github.com/Sillivan88/TSTableViewKit

Fazit

Dieser Artikel hat gezeigt, wie sich eine eigene Tabelle mit mehreren Reihen und Spalten umsetzen lässt, die dank Delegation dynamisch generiert und zusammengebaut wird. Natürlich kann man solch eine Klasse noch weiter ausbauen und um haufenweise weitere Funktionen ergänzen. So könnte man den Nutzer selbst wechselnde Hintergrundfarben für die Zellen festlegen lassen oder mehr Formatierungsoptionen für die Header und Standardtexte bereitstellen. Doch die gezeigten Schritte stellen bereits einen sehr guten Anfang dar, auf dem man aufbauen und in den man neue Methoden integrieren kann.

Darüber hinaus hat dieser Artikel die typische Entwicklung einer eigenen, aufwendigen View-Klasse aufgezeigt. Schritt für Schritt wurden alle notwendigen Funktionen integriert, um das gewünschte Verhalten zu erhalten. Zudem ist die Klasse so konzipiert, dass sie sehr gut noch erweitert werden kann, ohne dass ungewollte Abhängigkeiten bestehen. ■

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.
www.thomassillmann.de

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

NEO4J: UMSTIEG IN DIE REALE WELT DER GRAPHEN

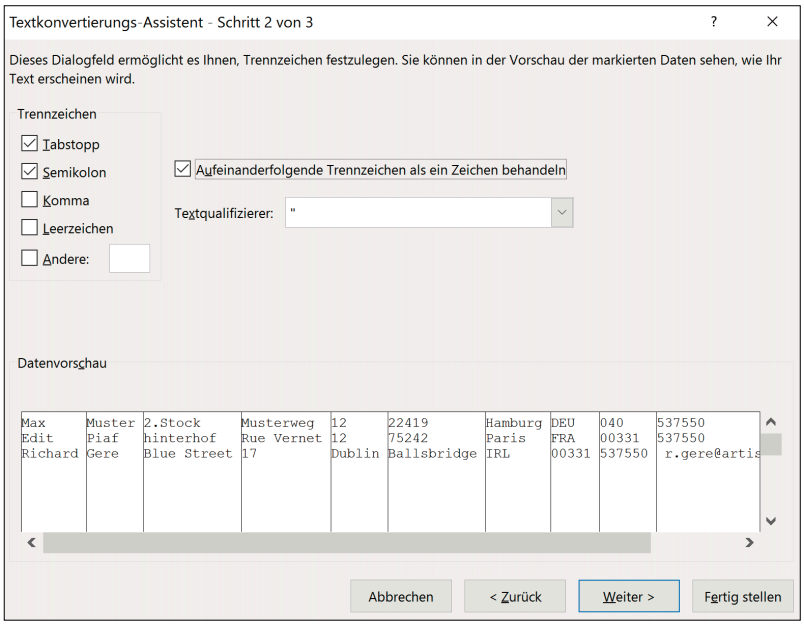
Komplexe Beziehungen

Neo4j besitzt komfortable Import-Schnittstellen, um vorhandene Datenbestände in eine Graph-Datenbank zu überführen.

Graphen unterstützen wesentlich besser als relationale Datenbanken die Nutzung komplexer, dynamischer Beziehungen. Die Fähigkeit von Graph-Datenbanken, eng miteinander verflochtene Daten leichter zu modellieren und einfacher zu analysieren, bringt Unternehmen Vorteile gegenüber ihren Mitbewerbern.

Im Unterschied zu den letzten Jahrzehnten, in denen atomare Daten im Fokus der Informationsverarbeitung standen, nimmt heutzutage das Erfassen, Analysieren und Verstehen von Beziehungen eine entscheidende Rolle ein. Dieser Trend verstärkte sich durch die Entwicklung des Internets zu einer ganzen Welt vernetzter Beziehungen im Web.

Nach wie vor befindet sich die weitaus überwiegende Mehrheit der Daten in relationalen Datenbanken, da diese Technik immer noch als der Industriestandard gilt. Inzwischen erkennen aber zunehmend mehr Anwender relationaler Systeme die diesen Datenbank-Management-Systemen (DBMS) inhärenten Beschränkungen.



Einige Anwendungen erlauben beim Import einer CSV-Datei die Definition ihres strukturellen Aufbaus, um so die Regeln für den Datenimport individuell festzulegen (Bild 1)

Die stürmische Entwicklung und Verbreitung sogenannter NoSQL-Datenspeicher in den letzten Jahren verdeutlicht diese Tendenz. Selbst große Hersteller relationaler Datenbanken wie Oracle oder IBM versuchen, die mit der NoSQL-Technologie verbundenen Vorteile in ihren eigenen DBMS zu nutzen. Jedoch lässt sich die mit den Beziehungen vernetzte Intelligenz optimal nur mit einer Graph-Datenbank umsetzen.

Ausgangsbasis ist die relationale Welt

Um sich als Graph-Datenbank am Markt zu etablieren, benötigt man als Grundvoraussetzung Schnittstellen, um vorhandene Datenbestände aus fremden Systemen zugänglich zu machen. Neo4j besitzt für diese Zwecke Import-Funktionen, um Daten aus anderen Datenhaltungssystemen für eigene Anwendungen im Datenbanksystem selbst zu nutzen.

Als Graph-Datenbank bietet Neo4j mit CQL (Cypher Query Language) eine leistungsfähige Datenbanksprache, mit der man Datenbestände auch für die Zielsetzungen des Prototyping importieren kann. Mit CQL realisiert man recht schnell die grundlegenden Datenstrukturen des Graphen und erhält über den Neo4j-Browser eine sofortige Rückmeldung zum gewünschten Erfolg der Datenübernahme.

Ein Datenimport aus dem relationalen System mittels Cypher und Neo4j-Browser eignet sich immer für die Zwecke des Prototyping – auch für den Import Hunderter oder Tausender Zeilen pro relationaler Tabelle. Diese Vorgehensweise erlaubt es, schnell einen Prototyp der Graph-Datenbank inklusive einer Anwendung zu implementieren. Damit gewinnt man vor allem wichtige Erkenntnisse über die Anforderungen der Fachabteilung und die zugrunde liegenden Strukturen in den relationalen Datenbeständen.

Implementierung verbessern

Diese dienen auch als Ausgangsbasis, um die Implementierung einer Graph-Datenbank zu verbessern. Beschränkt sich der Import auf nur wenige Hundert Zeilen aus den relationalen Tabellen, so reicht ein Einsatz der beiden Werkzeuge CQL und Neo4j-Browser aus. Die Übernahme Hunderttausender oder gar Millionen von Tabellenzeilen in eine Graph-Datenbank erfordert allerdings eine andere, komfortablere Strategie.

Diesen Zusammenhang hat Neo Technology als Hersteller von Neo4j erkannt und seit Version 2.1 eine eigenständige Schnittstelle für den Import von Daten implementiert. Diese

Schnittstelle basiert auf dem CSV-Format (Bild 1), kann im Online- oder Batch-Betrieb genutzt werden und ist auch innerhalb von CQL verfügbar.

Cypher kennt das Built-in-Kommando *LOAD CSV*, um Daten in einen Graphen zu importieren. Das *LOAD*-Kommando bietet verschiedene Optionen, um den Import der CSV-Datei an ihren individuellen Aufbau anzupassen. Es spielt keine Rolle, ob die CSV-Datei auf dem lokalen Computer oder einer Remote-Hardware gespeichert ist. Für diese Zwecke kennt der *LOAD CSV*-Befehl in Cypher unterschiedliche Zugriffsmethoden auf die CSV-Datei wie https, http, ftp oder file://.

Allerdings eignet sich der *LOAD CSV*-Befehl nicht für den Import von Massendatenbeständen mit mehr als 100 Millionen Zeilen in einer CSV-Datei. Dieses Cypher-Statement unterstützt um die eine Million Zeilen in einer CSV-Datei und arbeitet (entsprechende Wartezeit vorausgesetzt) auch noch mit mehr als zehn Millionen von Zeilen zuverlässig.

Ferner sollte man vor der Ausführung einige Voraussetzungen schaffen, damit das DBMS einen *LOAD CSV*-Import auch bei größeren Datenbeständen noch erfolgreich ausführt. Vor allem soll die Performance des *LOAD CSV*-Imports ein zufriedenstellendes Zeitverhalten besitzen.

Aufbereitung von CSV-Daten

Im letzten Jahrzehnt setzte sich das CSV-Format vor allem im Umfeld des Handels, der Industrie, der Banken, der Versicherungen, der Statistik- und Finanzämter immer stärker durch. Die parallel stattfindende Verbreitung des Internets und das Aufkommen neuer Anwendungen im Web verstärkten die Tendenz, CSV-Dateien als Austauschformat zu nutzen. Ausgehend von den Büroanwendungen mit integrierter Textverarbeitung entwickelten sich immer mehr kostenlose Anzeigeprogramme und Editoren für CSV-Dateien (Bild 2).

Diese Anwendungen zeigen nicht nur den Inhalt einer CSV-Datei an, sondern ändern deren Inhalte, ganz bequem ihre Struktur oder speichern sie in einem anderen Format.

Auch unterstützen diese CSV-Anwendungen das Zerlegen einer zusammengesetzten Spalte in ihre Einzelbestandteile, das Löschen, Sortieren oder Tauschen von Spalten/Zeilen in ihrer Reihenfolge, das Ändern des Trennzeichens oder der Zeichensätze. Manche Programme besitzen neben einer Kommandozeile sogar integrierte Schnittstellen zu gängigen Datenbanken (dBase, FoxPro, SQLite, MySQL), um direkt aus diesen eine CSV-Datei zu erstellen. Einige erlauben sogar, ausgehend von einer CSV-Datei, ein SQL-Skript – beispielsweise das erforderliche *CREATE TABLE*-Statement – zu erzeugen.

Damit generiert man SQL-Befehle für den Import in Datenbanken, die CSV-Dateien nicht direkt verarbeiten. Einen derartigen parametergesteuerten Generator für CQL gibt es derzeit noch nicht; man kann dies jedoch über ein Tabellenprogramm manuell durchführen.

Manche CSV-Editoren bieten das Zusammenfügen mehrerer CSV-Dateien zu einer gemeinsamen Datei an. Einige davon verfügen über Funktionen, um Datentypen – beispielsweise Datums-, Zeitangaben, Zeichenketten oder Aufzählungstypen – in ein anderes Format umzuwandeln.

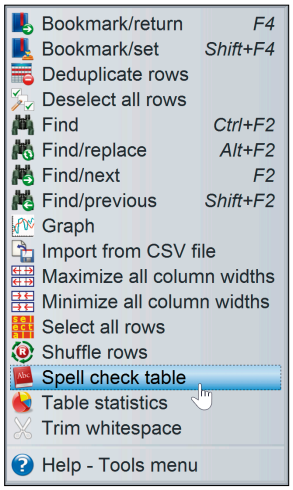
CSV-Dateiformat für den Datenaustausch

Das Dateiformat CSV (Comma-Separated Values, seltener Character-Separated Values) beschreibt den Aufbau einer Textdatei für die Speicherung oder den Austausch einfach strukturierter Daten.

Ursprünglich zielte das CSV-Format auf den Austausch oder die Konvertierung von Daten zwischen Tabellenverarbeitungen verschiedener Hersteller ab. Inzwischen nutzt man CSV, um Daten zwischen unterschiedlichen Computerprogrammen, vor allem Datenbanken, auszutauschen.

Stand heute gibt es keine Norm und keinen Standard für das CSV-Dateiformat; jedoch existiert seit über zehn Jahren der RFC 4180 als Request for Comments der Internet Engineering Task Force (IETF).

Die Bandbreite für Interpretationen einer CSV-Textdatei hält sich aber aufgrund ihres weitgehend einheitlich einfachen Aufbaus und den damit verbundenen Konventionen in Grenzen.



Für das Bearbeiten von CSV-Dateien bietet CSV Editor Pro mächtige Funktionen: Neben einem Filter zum Ausblenden von Teilen der CSV-Datei besitzt das Programm unter anderem einen Multi-Line-Edit-Modus, Lesezeichen und eine Rechtschreibprüfung (Bild 2)

Funktionen zum Vergleichen von CSV-Dateien

Funktionen zum Vergleichen mehrerer CSV-Dateien, was Struktur und Inhalte betrifft, oder gar deren automatische Anpassung an eine vorgegebene Struktur (am sinnvollsten im Hintergrundbetrieb) findet man recht selten. Somit besitzt keiner der CSV-Editoren einen wirklich umfassenden Komfort, nur wenige betrachten die Inhalte einer CSV-Datei und führen mit ihnen Berechnungen oder einen einfachen Vergleich auf Unterschiede durch. Die Notwendigkeit derartiger Anpassungen resultiert aus der immer noch fehlenden Standardisierung des CSV-Formats.

Hilfreich für Überarbeitungen der Formatdarstellungen erweist sich der Einsatz der Spaltenfunktionen einer Tabellenverarbeitung. Damit lassen sich ganze Spalten der CSV-Datei, was die Darstellung ihrer Inhalte betrifft, entsprechend in andere Zeichenketten umformen. Natürlich sind auch ▶

Berechnungen mittels der Feldfunktionen in der Tabellenverarbeitung möglich. Analog lässt sich während der Durchführung eines Imports über die Befehle *CREATE/MATCH/MERGE* mit den Cypher-Funktionen verfahren. So initialisiert man eventuell in den CSV-Feldern vorhandene Nullwerte durch die *COALESCE*-Funktion mit geeigneten Standardwerten.

Bei Bedarf lassen sich so die Daten über Cypher-Statements auch filtern, aggregieren oder in ihrer Darstellung verändern (zum Beispiel von Groß- in Kleinbuchstaben oder umgekehrt).

Dies kann sowohl für Labels oder Properties von Knoten als auch für Relationships erfolgen. In der CSV-Datei fehlende Felder, aber für die Graph-Datenbank gewünschte Properties oder Label erzeugt man über Cypher. Dazu eignet sich beim *MERGE*-Befehl beispielsweise die *ON CREATE*-Option, die Properties beim Anlegen eines Elements der Graph-Datenbank passend mit Werten belegt.

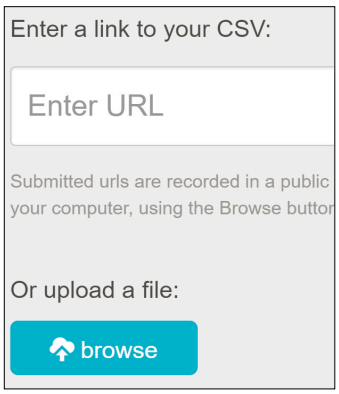
Kein einheitlicher Standard bei der Struktur von CSV-Dateien

Stand heute gibt es leider mangels Standards immer noch viele verschiedene Möglichkeiten, eine CSV-Datei aufzubauen und sie zu strukturieren.

Ebenso verhält es sich mit der Codierung der CSV-Datei, was Inhalt und Zeichen betrifft. Selbst nach Aufbereiten einer CSV-Datei kann sich diese bezüglich ihrer Struktur in einem inkonsistenten Zustand befinden. Insofern empfiehlt es sich, insbesondere beim Arbeiten mit vielen einzelnen CSV-Dateien oder mit CSV-Dateien großen Umfangs, diese vor dem eigentlichen Einsatz auf Fehler zu prüfen.

Für die Inspektion des CSV-Formats eignen sich in Kombination besonders die nachfolgenden Programme/Websites, da sie verschiedene Typen oder Klassen an Werkzeugen repräsentieren:

- Online CSV Conversion Tools: Eine von der Data Design Group realisierte Website, um CSV-Dateien in andere Dateiformate (HTML, JSON, SQL, XML, YAML) zu konvertieren. Auch die umgekehrte Richtung zur Erzeugung und Prüfung von CSV-Dateien ist realisiert.
- Papa Parse (früher: JQuery Parse Plugin): Eine JavaScript-Bibliothek, um CSV-Dateien im Webbrowser über HTML5 zu lesen und zu parsen. Der Entwickler Matt Holt hat den Quellcode unter MIT-Lizenz gestellt.
- CSVLint: Ein beim ODI Tech Team in Entwicklung befindliches Werkzeug (derzeit im Alpha-Status), das sowohl zur Validierung als auch über ein Webservice-API zur Einbettung in eine Website zur Verfügung steht.
- CSV Validator: Ein Werkzeug des TNA (The National Archives), dem Nationalarchiv von England, Wales und dem Vereinigten Königreich (außer Schottland und Nordirland). CSV Validator ist als Open-Source-Programm frei zugänglich.



CSVLint speichert per URL übergebene CSV-Dateien in öffentlich zugänglichen Testberichten; eine Übergabe der CSV-Datei mittels der *browse*-Schaltfläche verhindert dieses Verhalten (Bild 3)

Die Online CSV Conversion Tools stellen über clientseitiges JavaScript auf der Website zahlreiche Formatwandler bereit. Die Realisierung stammt von der Data Design Group, einem Softwarehaus mit umfangreichen Erfahrungen im Datenbank-Umfeld (speziell Oracle).

Vielfältige Einstellungen

Die Website bietet vielfältige Einstellungen für den strukturellen Aufbau einer CSV-Datei. Auch Sortieren, mehrzeilige Feldwerte, Auswahl der Trennzeichen und die Prüfung auf Korrektheit nach RFC 4180 sind implementiert.

Anhand einer CSV Template Engine mit Template Writer passt man den Output an individuelle Vorstellungen an. Als weiteres Feature generiert die Website anhand von Vorgaben beispielhafte Testdaten und erzeugt so CSV-Dateien für Massentests. Auch die Website von Mockaroo eignet sich für eine realistische Generierung von Testdaten.

Papa Parse lässt sich als JavaScript-Bibliothek in Webseiten über den Browser am Client integrieren, um eine CSV- in eine JSON-Datei (JavaScript Object Notation) zu überführen. Der Parser enthält Funktionen, um Fehler wie falsch gesetzte Anführungszeichen oder Trennzeichen in Aufbau und

Aufbau einer CSV-Datei

Als reine Textdatei orientiert sich der CSV-Dateityp an eingebürgerten Konventionen, um verschiedene Bereiche innerhalb der Datei voneinander zu unterscheiden.

- Datensätze: Als Zeichen für die Trennung einzelner Datensätze kommt in der Regel der Zeilenumbruch zum Einsatz.
- Datenfelder: Am häufigsten verwendet man das Komma, um die Datenfelder voneinander zu unterscheiden. Abhängig von durch die Software oder durch den Benutzer getroffenen Voreinstellungen sind auch Strich-, Doppelpunkt, Leerzeichen oder Tabulator üblich.
- Inhalt – die eigentlichen Daten: Hierzu greift man auf das Anführungszeichen als Feldbegrenzer zurück. Tritt dieses Sonderzeichen selbst in den Daten auf, so gibt man das Anführungszeichen einfach doppelt im Sinne eines Maskierungs- oder Escape-Zeichens an.
- Zusätzliche Regeln: Im Lauf der Zeit führte man weitere Regeln ein oder verkettete mehrere CSV-Dateien miteinander, um nicht nur einfache, sondern auch kompliziertere, beispielsweise geschachtelte Datenstrukturen verarbeiten zu können.

Struktur von CSV-Dateien zu entdecken und ihre Ursache zu analysieren. Die Software unterstützt Worker Threads und arbeitet so im Multi-Threaded-Mode, um eine Webseite mit hoher Arbeitslast noch responsive zu halten.

Als Besonderheit wandelt das Werkzeug auch JSON- in CSV-Dateien um. Ergänzend realisierte der Entwickler eine Online-Demo zum schnelleren Kennenlernen von Papa Parse. Mit BabyParse steht eine serverseitige, mit Node.js-kompatible Implementierung bereit.

Lesbarkeit der CSV-Dateien mit CSVLint testen

CSVLint testet zum einen die Lesbarkeit der CSV-Datei, zum anderen prüft das Tool zusätzlich, ob die CSV-Datei alle erforderlichen Spalten und Datentypen enthält. Die Überprüfung der CSV-Datei erfolgt unmittelbar beim Upload auf die Homepage von CSVLint; allerdings müssen die Datenwerte der CSV-Datei durch Begrenzer korrekt voneinander getrennt sein.

Zusätzlich verarbeitet CSVLint auch ZIP-Archive mit mehreren CSV-Dateien. Optional bindet das Tool ein CSV-Schema im JSON-Format bei der Prüfung ein, um weitergehende Tests zu ermöglichen. ▶

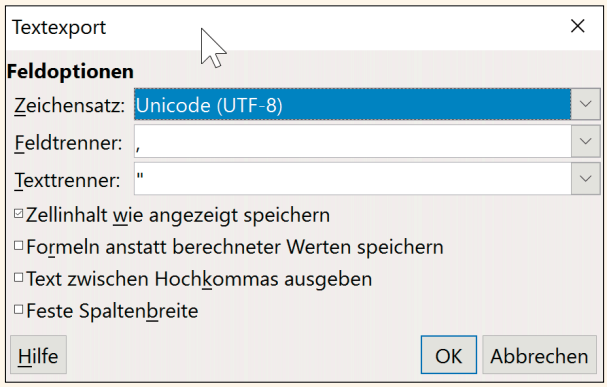
UTF-8-Codetabelle als Standard-Codierung für CSV

UTF-8 stellt aufgrund des De-facto-Standards der Zeichencodierung im Internet die am weitesten verbreitete Codierung für Unicode-Zeichen dar.

Für das CSV-Format bildet die Codierung der Datei im UTF-8-Zeichensatz den Schlüssel zum Erfolg.

Die UTF-8-Codierung unterstützt Zeichenketten bis zu einer Länge von vier Byte, mit denen sich alle Unicode-Zeichen abbilden lassen.

Sollen zusätzlich Zeichen aus dem Japanischen, Chinesischen, Koreanischen, Arabischen oder Russischen in der CSV-Datei auftreten, so greift man auf UTF-16/32 zurück. Um Inhalte von UTF-16/32-codierten Dateien auf dem Bildschirm anzuzeigen, benötigt man spezielle Unicode-Viewer und eine Installation der entsprechenden Schriftfonts.



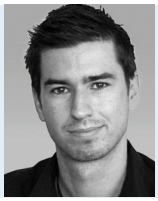
Viele Anwendungsprogramme bieten beim Speichern als CSV-Datei die UTF-8-Codierung an

developer media

developer media Webinare

Komprimiertes Know-how für Entwickler

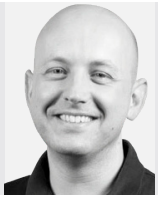
Mit WPF und PRISM Anwendungen entwickeln
Referent: Christian Giesswein
On-demand, 120 min.



Einführung in CQRS
Referent: Philip Jander
On-demand, 120 min.

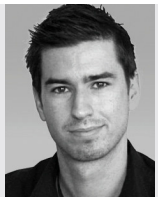
MS SQL Server – mehr als nur eine Datenbank
Referent: Thorsten Kansy
On-demand, 120 min.

Cross-Plattform-Entwicklung mit Visual Studio
Referent: André Krämer
On-demand, 120 min.



CQRS und Multi-Model-DB
Referent: Jan Fellien
On-demand, 60 min.

Entity Framework und C#
Referent: Christian Giesswein
On-demand, 60 min.



Smart Development
Referent: Alexander Schulze
On-demand, 60 min.

Weitere Informationen und Anmeldung unter
www.developer-media.de/webinare

Das Prüfergebnis teilt CSVLint in drei Klassen ein: Fehler, Warn- und Informationsmeldungen. Fehler müssen beseitigt werden, um den weiteren Verarbeitungsprozess einer CSV-Datei nicht zu stören.

Bei Warnmeldungen ist dies nicht erforderlich, ihre Beseitigung erleichtert lediglich die Weiterverarbeitung, ihr Vorhandensein wirkt sich aber nicht störend aus. Informationsmeldungen geben lediglich Hinweise oder Tipps und erleichtern so gegebenenfalls den Verarbeitungsprozess. Um den Inhalt der CSV-Datei zu schützen, darf deren Upload nicht über den URL, sondern sollte über die *browse*-Schaltfläche (Bild 3) erfolgen.

Das TNA realisierte große Teile des CSV Validators in Scala, sodass die Anwendung innerhalb der Java Virtual Machine (JVM) läuft. Damit lässt sich das Programm unter allen Betriebssystemen mit JVM einsetzen.

Als Alleinstellungsmerkmal basiert der CSV Validator auf einer CSV Schema Language, die als Grundlage zur Spezifikation eines CSV Schemas dient. Damit kann man neben der Struktur (dem Aufbau) und den Datentypen auch Regeln für die CSV-Datei vorgeben. Die Anwendung besitzt sowohl eine Kommandozeile als auch eine grafische, mit Swing realisierte Bedienoberfläche. Für die Programmierung stehen eine Scala- sowie eine Java-Bibliothek zur Einbindung in eigene Projekte zur Verfügung. Ebenfalls empfehlenswert ist die seit Kurzem in Version 1 verfügbare csvkit-Toolsuite.

Vorgehensweise des Datenimports

Daten in einer Graph-Datenbank besitzen in der Regel vielfältige Beziehungen. Die Wahrung der Konsistenz des Graphen gewährleistet das DBMS über die Definition von Integritätsbedingungen. Generell muss sich in einer Welt inhärenter Beziehungen zwischen den Knoten die Aufnahme neuer Daten an dessen Graphmodell orientieren.

Solange sich noch kein Graph in der Datenbank befindet, gestaltet sich der Import von Daten sehr einfach: Es gibt noch keine dem DBMS bekannten Integritätsbedingungen, die bei der Aufnahme neuer Daten in den Graphen einzuhalten sind. Natürlich kann eine Beziehung immer nur zusammen mit den beteiligten Knoten oder mit bereits in der Graph-Datenbank existierenden Knoten importiert werden.

ODI – das Open Data Institute

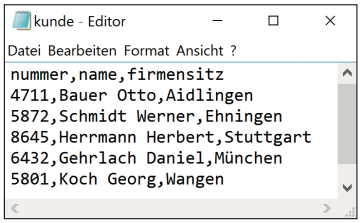
Das als Non-Profit-Firma von Tim Berners-Lee (dem Erfinder des World Wide Web) und Nigel Shadbolt gegründete ODI zielt darauf ab, die Open-Data-Kultur weltweit voranzubringen.

Die Finanzierung des ODI erfolgt über das Technology Strategy Board (Innovate UK); in Deutschland verfolgen das BMBF (Bundesministerium für Bildung und Forschung) und das BMWi (Bundesministerium für Wirtschaft und Energie) nur schleppend derartige Aufgaben.

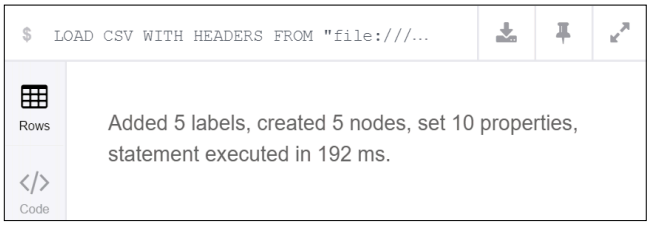
Daten nehmen eine enorm wichtige Stellung in der Gesellschaft ein; für unterschiedliche Organisationen besitzen sie sowohl einen wirtschaftlichen, sozialen als auch ökologischen Wert.

Seit der Gründung des ODI in Großbritannien breitete sich die Erkenntnis über den Wert von Daten weltweit aus, sodass ähnliche Organisationen auf der ganzen Welt entstanden.

Das ODI bietet Weiterbildung an, betreibt Forschung anhand zweier Projekte: DaPaaS (Data and Platform as a Service) und OpenDataMonitor, und unterstützt Firmengründungen, die zu einer weiteren Verbreitung der Open-Data-Plattform beitragen sollen.



Die CSV-Datei kunde.csv enthält für das Label *kunde* in der Header-Zeile die Properties *nummer, name* und *firmensitz* (Bild 4)



Der LOAD CSV-Befehl hat fünf Knoten mit dem *kunde*-Label und jeweils zwei Properties (insgesamt also 10 Datenwerte) in die Graph-Datenbank importiert (Bild 5)

Integritätsbedingungen

Gibt es schon Daten in der Graph-Datenbank, so basieren diese auf einem Graphmodell mit eventuell definierten Integritätsbedingungen. Beim Import von Daten in eine bestehende Graph-Datenbank muss man sich überlegen, welche Vorgehensweise am sinnvollsten ist.

Es gilt sicherzustellen, dass vorhandene Integritätsbedingungen nicht den Datenimport behindern. Insbesondere ist zu entscheiden, welchen Knoten man als Startpunkt für die Neuaufnahme von Daten wählt.

Ein analoger Sachverhalt gilt für den Endknoten, also den Knoten, der als letztes Element in die Graph-Datenbank hinzugefügt wird. Für die Elemente zwischen Start- und Endknoten muss man eine geeignete, sukzessive Vorgehensweise wählen, sodass keine Inkonsistenzen in der Graph-Datenbank auftreten.

Das Graphmodell einer Neo4j-Datenbank implementiert für die Konsistenz derzeit sowohl Eindeutigkeit als auch Existenz auf einem recht einfachen Niveau. So bietet Neo4j in der aktuellen Version keine Wertebereichsintegrität durch Vorgabe gültiger Werte (zum Beispiel Ober- und Untergrenzen). Zudem unterstützt das DBMS diese Konsistenzprüfungen immer auf der Ebene von Properties. Properties kann man in der Graph-Datenbank für Knoten und Beziehungen definieren.

Eindeutigkeit und Existenz für Properties stellt das DBMS derzeit nur für Knoten als Elemente des Graphen sicher. Für Beziehungen, die das Graphmodell als Kanten zwischen Knoten abbildet, bietet Neo4j in der Enterprise-Edition zusätzlich noch Property-Existence-Constraints an. Über diese Art von Integritätsbedingung verfügt die Community-Edition von Neo4j nicht.

Während ein Unique-Constraint die Eindeutigkeit für ein Property eines bestimmten Labels im Graphmodell sowohl für Knoten als auch für Beziehungen sicherstellt, gewährleistet ein Property-Existence-Constraint lediglich dessen Existenz. Einen Import von Daten führt das DBMS im Fall des Unique-Constraints nur bei Eindeutigkeit der Property für das Label durch.

Im Unterschied hierzu prüft die Integritätsbedingung der Existenz, ob die Property für alle Knoten mit einem bestimmten Label oder für alle Kanten (Beziehungen) mit einem bestimmten Typ in der Graph-Datenbank vorhanden ist. Ist diese Bedingung erfüllt, so gewährt das DBMS die Durchführung des Datenimports; im anderen Fall wird der Import abgelehnt.

Daten mittels CSV über Cypher importieren

Beim Import von Daten mittels des CSV-Formats wählt man ein Element der Graph-Datenbank (Kante/Knoten) gemäß der diskutierten Vorgehensweise aus. Das so bestimmte Element entspricht einem Label der Graph-Datenbank, das über zusätzliche Properties verfügt. Es empfiehlt sich, die CSV-Datei auf Ebene des Betriebssystems mit dem Format CSV zu verknüpfen.

Als Bezeichnung/Namen für die CSV-Datei wählt man den Namen des gewählten Labels. Dann kann direkt das betroffene Element der Graph-Datenbank aus dem Namen der CSV-Datei abgelesen werden. Es empfiehlt sich, als ersten Eintrag in der CSV-Datei eine sogenannte Header-Zeile aufzunehmen; in dieser ersten Zeile gibt man die Namen der Properties des jeweiligen Labels direkt an.

Handelt es sich beispielsweise um das Label *Kunde*, das eine Nummer, einen Namen und einen Firmensitz als Property besitzt, so führt man diese Metadaten in der ersten Zeile der CSV-Datei auf. Im Beispiel besitzt die CSV-Datei den Namen *kunde.csv* mit dem Eintrag *nummer, name, firmensitz* in der ersten Zeile (Bild 4).

Unterhalb dieser ersten Header-Zeile folgen in der CSV-Datei sämtliche Ausprägungen des Labels in der durch die erste Zeile vorstrukturierten Listenform. Das heißt, jede Ausprägung eines derartigen Elementtyps der Graph-Datenbank bildet eine Zeile in der CSV-Datei. So steht der erste Kunde mit den Attributausprägungen *4711, Otto Bauer, Aidlingen* in der zweiten Zeile der CSV-Datei.

Anhand der durch die Metadaten (Header-Zeile) vorgegebenen Reihenfolge der Spalten befinden sich sämtliche Ausprägungen der Kundendaten als Zeileneinträge. Eine Ausprägung eines Zeileneintrags entspricht einem Element mit seinen Datenwerten, also den zugehörigen Werten der Properties, des aktuell ausgewählten Labels.

Das Komma dient als Trennzeichen zwischen den Datenwerten einer einzelnen Zeile; es trennt die Werte der Properties des Labels voneinander ab. Für ein Prototyping des CSV-Imports eignet sich der Neo4j-Browser, den man über den URL http://localhost:7474/ bei gestartetem DBMS erreicht. Mit einem *LOAD CSV*-Befehl, der auf die CSV-Datei verweist, importiert man die dortigen Zeileneinträge in die Graph-Datenbank.

Ein hilfreicher Cypher-Befehl für den CSV-Import

Die Übernahme der Zeilen einer CSV-Datei mit dem *LOAD CSV*-Befehl erfolgt mittels des nach dem *AS*-Parameter folgenden Cypher-Statements. Dabei handelt es sich in der Regel um einen *CREATE*- oder *MERGE*-Befehl; beide Befehlstypen setzt man analog der gängigen Vorgehensweise beim sukzessiven, schrittweisen Aufbau einer Graph-Datenbank ein. Beim *LOAD CSV*-Befehl fungiert der mittels *AS*-Para- ▶

Im Zentrum steht das Modell des Graphen

Im Unterschied zu anderen Datenbanken besitzt Neo4j kein externes Schema, das Aufbau und Struktur der Graph-Datenbank beschreibt.

Vor jedem Datenimport in eine Graph-Datenbank sollte man sich immer ausführlich Gedanken über ein passendes Graphmodell machen.

Eine einfache Übernahme des Datenmodells aus der relationalen Welt führt meistens nicht zu einem geeigneten Graphmodell.

Zielführender hat sich eine Modellierung des Graphen ausgehend von möglichst vielen Szenarien über dessen fachlichen Einsatz in Anwendungen gezeigt. Diese Vorgehensweise hilft auch, die Graph-Datenbank flexibler (für später noch zu entwickelnde Anwendungen) zu gestalten.

Integritätsbedingungen bei Neo4j

Als DBMS realisiert Neo4j derzeit nur einen Typ von Integritätsbedingung: Constraints.

Von diesen Constraints kennt Neo4j drei verschiedene Arten: Unique-Node-Property-, Node-Property-Existence- und Relationship-Property-Existence-Constraints; jeweils bezogen auf ein bestimmtes Label.

Alle drei Arten von Constraints definiert man in Cypher über den *CREATE CONSTRAINT ON*-Befehl.

Derzeit unterstützt Neo4j keine Kardinalität von Beziehungen; es kann nur die Eindeutigkeit oder deren Existenz für ein vorzugebendes Label gewährleistet werden.

Achtung: Eine Datenbank der Enterprise Edition mit Property-Existence-Constraints kann mit der Community-Edition nicht gelesen, geschweige denn bearbeitet werden.

meter übergebene Name, beispielsweise *csvZeile*, als externe Referenz auf die Zeile der CSV-Datei. Über ihn greift man bei der Formulierung des *CREATE/MERGE*-Befehls auf die externen Inhalte der Zeileneinträge der CSV-Datei zurück. Der beim *AS*-Parameter genannte Bezeichner übernimmt für das jeweilige Label die zugehörigen Inhalte der Zeileneinträge aus der CSV-Datei.

Konvertierungsfunktionen von Neo4j 2

Verwendet man die *WITH HEADERS*-Option beim *LOAD CSV*-Befehl, so greift man über deren Namen auf die tatsächlich benötigen Spalten zu und muss nicht alle auswählen. Ohne Header-Zeile spricht man die erste Spalte der CSV-Datei über den Feldindex [1] oder die n-te Spalte über *csvZeile[n]* an. Im Beispiel der *kunde.csv*-Datei findet der Zugriff anhand des *AS*-Parameters *csvZeile* auf den zugehörigen Datenwert statt.

So dient die Referenz *csvZeile.name* im *CREATE/MERGE*-Befehl dazu, das Label *kunde* und die Property *name* in der Graph-Datenbank mit dem zugehörigen Wert zu versehen. Da die Daten der CSV-Datei als Zeichenketten vorliegen, wendet man im Fall von Zahlen die gängigen Konvertierungsfunktionen von Neo4j an.

Es empfiehlt sich, bei einer großen Menge an Datensätzen (mehr als zwanzigtausend), den Cypher-Befehl *USING PERIODIC COMMIT zeilenanzahl* zu nutzen. Standardmäßig setzt Cypher (bei nicht vorhandener *zeilenanzahl*) nach jeweils 1000 Zeilen einen Commit in Neo4j ab.

Diese automatische Commit-Ausführung (nach der jeweiligen Anzahl von Zeilen) verhindert Speicherfehler, die seitens des DBMS oder der JVM eventuell beim Überschreiten von Grenzen auftreten könnten.

Indizes und Unique Constraints

Zudem sollte man vor der Ausführung eines *LOAD CSV*-Befehls alle geplanten Indizes und Unique Constraints für die Graph-Datenbank definieren. Beides zusammen beschleunigt den *LOAD CSV*-Import, sobald man bei dessen Ausführung betroffene Elemente in der Graph-Datenbank mittels *MERGE* oder *MATCH* lokalisieren muss. Der passende *CREATE*-Befehl zur Aufnahme aller Zeileneinträge der *kunde.csv*-Datei (Bild 5) für die ersten beiden Spalten lautet:

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS
FROM "file:///E:/Neo4j-Daten/import/kunde.csv"
AS csvZeile
CREATE
(k:Kunde { nummer: TOINT(csvZeile.nummer),
name: csvZeile.name });
```

Die Durchführung eines CSV-Imports in Cypher mittels des *LOAD CSV*-Befehls gestaltet sich in der Praxis als recht einfach – vorausgesetzt, man kennt die Zusammenhänge und beachtet einige der nachfolgenden Ratschläge.

Insgesamt sollte dieser Import nicht mehr als 10 Millionen Knoten/Beziehungen umfassen; es sollte sich also um keinen Massenimport handeln.

Prüfung per Stichprobe

Vor einem tatsächlichen Import einer CSV-Datei ist es immer sinnvoll, die zugehörigen Daten auf ihre Fachlichkeit zu prüfen. Dazu eignet sich zum einen ein Test eines Ausschnitts der Daten (Stichprobe) auf fachliche und zum anderen eine Prüfung auf formale Korrektheit.

Beide Tests führt man mittels zweier *LOAD CSV*-Befehle und dazu geeigneten Parameter-Optionen durch. Um einen Ausschnitt der Daten auf Fachlichkeit und Durchführung des *LOAD CSV*-Befehls zu prüfen, verwendet man den nachfolgenden Cypher-Befehl mit Header-Klausel:

```
LOAD CSV WITH HEADERS
FROM "file:///E:/Neo4j-Daten/import/kunde.csv"
AS csvZeile WITH csvZeile
```

Aufbau und Strukturierung einer CSV-Datei

Für den CSV-Import von Neo4j verwendet man eine UTF-8-Codierung.

Das Zeichen für das Zeilenende bestimmt das jeweilige Betriebssystem.

Als Trennzeichen für Felder kommt standardmäßig das Komma zum Einsatz.

(Doppelte) Anführungszeichen können die Werte der Datenfelder zur besseren Lesbarkeit voneinander abheben. Beim Datenimport in die Graph-Datenbank entfallen diese Anführungszeichen, sie werden also nicht berücksichtigt.

Als Maskierungs- oder Escape-Zeichen verwendet man \ (Backslash); es schaltet die Bedeutung des danach folgenden Sonderzeichens ab.

Der LOAD CSV-Befehl für den Import von CSV-Daten

Cypher verfügt über einen flexiblen LOAD CSV-Befehl, um den Inhalt einer CSV-Datei als Elemente in eine Graph-Datenbank zu übernehmen.

Die Flexibilität des *LOAD CSV*-Befehls begründet sich in der Vielzahl von Parametern und Einstellmöglichkeiten.

So unterstützt dieser Cypher-Befehl für die CSV-Datei unterschiedliche Ablageorte (lokales Dateisystem, Netzwerk, Internet) durch verschiedene Zugriffsprotokolle file, ftp und URL (HTTP, HTTPS). Die Angabe des Ablageorts erfolgt nach der *FROM*-Klausel im *LOAD CSV*-Befehl.

LOAD CSV kennt verschiedene komprimierte Dateiformate wie gzip, Deflate oder ZIP. Der *LOAD CSV*-Befehl importiert CSV-Dateien mit Header-Zeile über die *WITH HEADER*-Option.

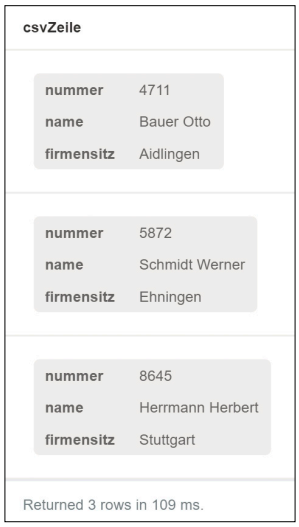
Über die Option *FIELDTERMINATOR* passt man das Standardzeichen für die Trennung der Datenfelder an die CSV-Datei an.

```
RETURN csvZeile
LIMIT 3;
```

Die Header-Klausel greift auf die erste Zeile der CSV-Datei zu und nimmt die dort vorhandenen Einträge als Bezeichnung für den Attributnamen in die Ergebnisliste auf. Eine Angabe dieser Attributbezeichner erlaubt es, die zugehörigen Attributwerte auf fachliche Korrektheit zu prüfen: Nach Ausführung des zuvor genannten *LOAD CSV*-Befehls gibt der Neo4j-Browser die ersten drei Zeilen der CSV-Datei in Tabellenform aus (Bild 6).

Die formale Korrektheit überprüft der nachfolgende Cypher-Befehl. Er testet primär, ob sich die Daten der CSV-Datei lesen lassen, zählt die Anzahl der Datensätze und gibt diese als Summe im Neo4j-Browser aus:

```
LOAD CSV
FROM
"file:///E:/Neo4j-Daten/import/kunde.csv"
AS csvZeile WITH csvZeile
RETURN COUNT(*);
```



Die Überprüfung der fachlichen Korrektheit der Daten einer CSV-Datei erfolgt im Neo4j-Browser mittels des *LOAD CSV*-Befehls (Bild 6)

Der Test auf fachliche Korrektheit der Daten beschränkt sich auf eine Stichprobe, schließt also keine vollkommene Fehlerfreiheit aus. In Cypher wäre für den *LOAD CSV*-Befehl eine spezielle Option oder Klausel sinnvoll, um für diesen Test die Elemente der Stichprobe tatsächlich auch zufällig auszuwählen. Damit würde Cypher aus der CSV-Datei nicht einfach nur die ersten Sätze gemäß dem Stichprobenumfang lesen, sondern die Auswahl wirklich rein zufällig vornehmen. Im Unterschied zum zweiten Test, der keine Stichprobe, sondern alle Sätze der CSV-Datei verarbeitet, garantiert dieser die vollständige formale Korrektheit aller Inhalte.

Als Entwickler oder Datenbankadministrator muss man die Bedeutung dieser beiden Tests und den damit verbundenen Nutzen kennen. Verzichtet man vollkommen auf die Überprüfung des Tests auf Fachlichkeit oder formale Korrektheit, so erhöht dies die Wahrscheinlichkeit des Vorhandenseins von Fehlern in der CSV-Datei.

Diese Fehler resultieren zum einen aus dem Aufbau, der Strukturierung, der Daten-Codierung oder dem mangelhaften Einsatz von Konvertierungsfunktionen. Insbesondere der erforderliche Einsatz der Konvertierungsfunktionen, um die Zeichenketten der CSV-Datei in Zahlentypen von Neo4j zu wandeln, erhöht die Wahrscheinlichkeit von Fehlern.

Leider zeigt sich die Korrektheit der Konvertierungsfunktionen letztendlich erst beim tatsächlichen CSV-Import in die Graph-Datenbank. Nimmt man vor dem endgültigen CSV-Import entsprechende Tests für die Konvertierungsfunktionen vor, so erspart man sich einen nicht unerheblichen Zeitaufwand für die endgültige Durchführung.

Als Ausweg aus diesem Dilemma empfiehlt es sich, die Umsetzung des CSV-Imports in einem ersten Schritt nur mit einem kleineren Umfang an Testdaten vorzunehmen. Lässt sich eine CSV-Datei mit wenigen Testdaten, also geringem Umfang an Sätzen, korrekt in eine Neo4j-Datenbank importieren, dann erledigt sich der zweite, nachgelagerte Schritt für den vollständigen CSV-Import sehr einfach durch Erweiterung der Datensätze.

Systemparameter für Import von Massendaten

Die Systemgrenzen von Neo4j für den Import von Massendaten (Bulk Data Imports) resultieren aus der Java-Technologie und aus der aktuell eingesetzten Hardware.

Primär muss man sicherstellen, dass die Hardware für den Import von Massendaten ausreichend Hauptspeicher zur Verfügung stellt. Die Hardware sollte 4 oder mehr Gigabyte freien Speicherplatz besitzen.

Zudem sollte man die Einstellungen der beiden Neo4j-Konfigurationsdateien für den Heap der JVM und das Memory Mapping des jeweiligen Datenbank-Page-Caches passend wählen:

- neo4j-wrapper.conf: wrapper.java.initmemory, wrapper.java.maxmemory (am besten beide gleich groß wählen, um ein Unterbrechen des Garbage-Collection-Zyklus zu vermeiden)
- neo4j.properties: neostore.nodestore.db.mapped_memory, neostore.relationshipstore.db.mapped_memory, neostore.propertystore.db.mapped_memory, neostore.propertystore.db.strings.mapped_memory, neostore.propertystore.db.arrays.mapped_memory

Empfehlungen für den Import von Massendaten

Unabhängig von der Datenbanktechnologie (Netzwerk, Hierarchie, Relational Datenbank, Graph) muss man für einen Massenimport immer Überlegungen zum Aufwand an Ressourcen (Zeit, Hauptspeicher) vornehmen. Im Unterschied zum Import wenig Tausender Datensätze gestaltet sich der Import mehrerer Hundert Millionen von Datensätzen wesentlich aufwendiger.

Es gilt diesen Aufwand an Ressourcen zu minimieren – nur dies gewährleistet die Einhaltung eventuell vorhandener Grenzen seitens des DBMS. Überschreitet ein Import vorgegebene Systemgrenzen der Datenbank oder des DBMS, so kann dies den Import als Ganzes verhindern. Hinzu kommen eventuelle Grenzen seitens der dem DBMS zugrunde liegende Basis-Software (Betriebssystem, Entwicklungswerkzeuge), die es ebenfalls zu berücksichtigen gilt.

Um einem Misslingen entgegenzuwirken, implementieren die Herstellers einer Datenbank für die Durchführung eines Massenimports (Bulk Data Import) zusätzliche Werkzeuge. Diese bezeichnen die Hersteller häufig als Dienstprogramme des DBMS; sie berücksichtigen bereits eigene Empfeh- ▶

lungen für die Vorgehensweise beim Massenimport. Damit arbeiten diese Dienstprogramme beim Bulk Data Import wesentlich schneller und zuverlässiger als die seitens des Anwenders geplante oder vorgenommene Vorgehensweise.

Im Fall der Graph-Datenbank Neo4j heißt das Import-Werkzeug Import Tool; manchmal wird es auch CSV Importer oder Neo4j-Import Tool genannt. Seit Version 2.2 beinhaltet Neo4j dieses speziell für den Massenimport entwickelte Werkzeug (Bild 7).

Dieses Tool führt den Import von Massendaten massiv parallel und höchst skalierbar durch, um eine ausreichende Performance zu erreichen.

Batch Insertion Facility

Das Import Tool darf nicht mit der im Neo4j-Kernel enthaltenen Batch Insertion Facility, einem weiteren Import-Werkzeug für Massendaten, verwechselt werden. Bei der Batch Insertion Facility handelt es sich um das in Java implementierte BatchInserter-Interface.

Mit diesem Interface lassen sich über ein Java-Programm Massendaten in eine neue Graph-Datenbank einfügen. Das Interface kann jedoch auch Daten in eine bereits vorhandene Graph-Datenbank übernehmen; allerdings muss dazu vor Ausführung des Java-Programms die Datenbank heruntergefahren werden. Die Batch Insertion Facility wiederum besitzt einige Besonderheiten, die man kennen muss, um mit ihr korrekt zu arbeiten:

- Not thread safe: Die Batch Insertion-Facility muss immer innerhalb eines einzigen Threads eingesetzt werden, oder der Zugriff darauf muss synchronisiert erfolgen, sodass zu einem Zeitpunkt nur ein Thread die Batch Insertion-Facility benutzt.
- Non-transactional: Transaktionen sind der Batch Insertion-Facility völlig unbekannt.
- Not enforcing constraints: Die Datenbank überwacht während der Ausführung der Batch Insertion-Facility keine ihr bekannten, das heißt vordefinierten Constraints.
- No index population: Die Datenbank baut keine Index-Strukturen auf und aktualisiert auch keine bestehenden.
- Shutdown: Die Datenbank befindet sich nach Beendigung der Batch Insertion-Facility in einem inkonsistenten Zustand. Erst ein Herunterfahren der Datenbank mittels *shutdown* baut Index-Strukturen auf oder aktualisiert diese und überprüft nachgelagert die Einhaltung vordefinierter Constraints.

Das CSV Import-Tool geht von gewissen Voraussetzungen aus, die erfüllt sein müssen, um mit Massendaten möglichst effizient zu arbeiten. Sind diese Konventionen nicht eingehalten, so muss man entweder durch Ändern der CSV-Dateien oder beim Start des Import-Tools dessen Übergabeparameter entsprechend vorbelegen.

Korrekter Import von Massedaten

Nur eine vollständige Einhaltung der gemachten Vorgaben für den Tool-Einsatz ermöglicht und garantiert einen korrekten Import vorliegender Massendaten. Primär bietet das Tool nur einen Import in eine leere Graph-Datenbank. Für alle anderen Fälle kommt der Batch Importer von Michael Hunger (einem Mitarbeiter von Neo Technology) als sinnvolle Alternative in Frage.

Aus Gründen einer geeigneten Performance für den Import massenhafter Daten unterstützt er keine Indizes; vielmehr muss man diese erst in einem nachgelagerten Schritt aufbau-

Testdaten für erfolgreichen CSV-Import

Vor der (letztendlichen) Durchführung eines CSV-Imports mit Massendaten benötigt man realistische Datensätze für vorausgehende Testdurchläufe.

Kennt man die zu importierenden Datentypen, so erzeugt man sich Testdaten mit Hilfe eines Daten-Generators, zum Beispiel über die Website Online CSV Conversion Tools.

Testdaten-Generatoren verfügen in der Regel über eine Funktion für den CSV-Export.

Schlüsselwörter spezifizieren die Generierung der erforderlichen Datentypen, diese sind im Idealfall auf die Fachlichkeit ausgerichtet und erzeugen so plausible realitätsnahe Daten.

Allowable Keywords

Keyword	Description
age	person's age 1 to 120
alpha alpha(n)	string of letters a to z mixed cas and if n specified then exactly n
birthday	date of birth in mm/dd/yyyy form
bool	true or false
char	1 single character of a letter or
city	cities
ccnumber	Credit Card Number
date date(2) or date(3) or date(4)	date in mm/dd/yyyy format or date(2) for dd/mm/yyyy or date(3) for yyyy/mm/dd or date(4) for yyyymmdd

Schlüsselwörter steuern die Generierung der Datentypen

Normalisierung von relationalen Datenbanken

Normalisierung einer relationalen Datenbank vermeidet Redundanzen und verhindert Anomalien im Datenbestand.

Die Informatik unterscheidet verschiedene Grade oder Formen der Normalisierung.

Eine Normalisierung kann man weitestgehend automatisiert mittels Algorithmen durchführen.

Relevant für die Praxis sind insbesondere die ersten drei Normalformen sowie die BCN (Boyce-Codd-Normalform).

In der Regel verzichtet man in der Praxis auf die Einhaltung der vieren und fünften Normalform aufgrund von Überlegungen zur Performance.

```
Function        Restart-Neo4jServer
Function        Set-Neo4jSetting
Function        Start-Neo4jArbiter
Function        Start-Neo4jBackup
Function        Start-Neo4jImport
Function        Start-Neo4jServer
Function        Start-Neo4jShell
Function        Stop-Neo4jArbiter
Function        Stop-Neo4jServer
```

Das Neo4j Import Tool steht (mit Ausnahme von Windows) als eigenständiges Programm zur Verfügung. Unter Windows ist das Import Tool als Neo4j-Management-Funktion innerhalb des Windows PowerShell Moduls von Neo4j realisiert (Bild 7)

```
Node counts
[>:??|*COUNT:76,29 MB---
Done in 31ms
Relationship counts
[*>:??------------------
Done in 11ms

IMPORT DONE in 1s 866ms.
  10 nodes
  5 relationships
  30 properties
```

Der Output des CSV Import-Tools (hier ein Ausschnitt davon) liefert zusätzliche Informationen über dessen Durchführung (Bild 8)

en. Zudem müssen CSV-Dateien die UTF-8-Codierung aufeisen und alle dasselbe Trennzeichen für die Felder (standardmäßig das Komma) verwenden.

Für Knoten und Relationships legt man getrennte CSV-Dateien jeweils mit speziellen Header-Zeilen an. Das Import-Tool erlaubt es, CSV-Datenquellen über mehrere Dateien zu verteilen, was sich im Fall von Massendaten immer als vorteilhaft erweist.

String als Standardbelegung

Die Header-Zeile jeder einzelnen CSV-Datei beschreibt über folgendes Muster *<name>:<feldtyp>* sowohl für Knoten als auch für Relationships einen Property-Namen mit zugehörigen Datentyp. Fehlt die Angabe eines Feldtyps, so greift das Import-Tool auf die Zeichenkette *(string)* als Standardbelegung zurück.

Als Feldtypen sind alle in Neo4j/Cypher verfügbaren Datentypen möglich; zudem verwendet man für Knoten noch IDs und *LABEL*. Relationships baut man über den Feldtyp *TYPE* für den Namen der Beziehung zusammen mit *START_ID* und *END_ID* für die beiden an der Beziehung beteiligten Knoten und eventuellen Properties auf.

Die in den CSV-Dateien für alle Knoten der Graph-Datenbank verwendeten IDs müssen für den gesamten Import-Prozess eindeutig sein; auch im Fall von Knoten mit verschiedenen Label. Der korrekte Aufbau der Relationships erfordert diese ID-Eindeutigkeit, nur dann findet das Import-Tool die richtigen, an der Beziehung beteiligten Knoten.

Ist die Eindeutigkeit nicht für alle Knoten, sondern nur für Knoten desselben Typs erfüllt, so kommt ein sogenannter ID Space zum Tragen. Innerhalb der CSV-Datei für einen Knoten, zum Beispiel *mitarbeiter.csv*, definiert man die ID dann über *ID(Mitarbeiter)* oder *ID(Firma)* als ID Space. In diesem Fall referenziert man Start- und Endknoten einer Relationship in der zugehörigen CSV-Datei mittels *START_ID(Firma)* beziehungsweise *END_ID(Mitarbeiter)*.

Der Einsatz des CSV Import-Tools erfolgt auf der Ebene des Betriebssystems; die eigentliche DBMS-Software darf dazu nicht gestartet sein. Viele Parameter und Optionen des CSV Import-Tools steuern dessen Ausführung über die Kommandozeile. Sogar das Verhalten des Werkzeugs im Fehlerfall bei den Daten oder der Header-Zeile lässt sich spezifizieren; aber auch Vorgaben für Optimierungen seitens der Prozessoren oder der Datenbankkonfigurationen lassen sich treffen.

Während eines laufenden Imports gibt das CSV Import-Tool zusätzliche Informationen und Statistiken aus (Bild 8). Damit zieht man Rückschlüsse über das spätere Laufzeitverhalten der Datenbank und eventuell erforderliche Tuning-Maßnahmen. Im Fehlerfall erhält man bei erneuter Ausführung über die Befehlsoption *--stacktrace* weitergehende Debug-Informationen.

Direkte SQL-Kopplung

Die Migration von Daten direkt aus einer relationalen Datenbank in einen Graphen stellt man sich in der ersten Betrachtung als recht einfach vor. Das mag abhängig von der Größe und der Semantik einer Datenbank aus Entwicklersicht mehr oder minder zutreffen. Seitens der Datenbank-Administration gestaltet sich diese Datenübernahme mit zunehmendem Anteil an der Ablösung des SQL-Systems entsprechend komplizierter. Zum einen müssen nicht nur die vorhandenen Strukturen der Daten und ihre Inhalte, sondern auch deren Semantik und enthaltenen Beziehungen sowie die vorhandenen Abfragen unter Umständen vollständig in die Welt der Graphen überführt werden. Schlussendlich stellt sich noch die Frage, wie mit den seitens der Datenbank-Administration implementierten Verfahren umgegangen werden soll. ▶

Probleme mit Zeichensätzen von Textdateien

Bei korrekt erfolgtem Filetransfer zwischen verschiedenen Betriebssystemen bleibt zwar deren Codierung erhalten, allerdings setzen Anwendungen unter Umständen eine spezifische Codierung voraus.

Hinzu kommt, dass Java die Byte Order Mark (BOM) in UTF-8-Texten nicht automatisch entfernt.

Die Anzeige der Zeichen am Bildschirm erfolgt nur korrekt, wenn die seitens der Anwendung erwartete Voreinstellung der Codierung auch in der Textdatei vorliegt.

Insofern kann es für eine Textdatei in UTF-8-Codierung unter Windows sinnvoll sein, diese in die für das Land und seine Sprache gängige Codierung umzuwandeln.

Für die sogenannten westlichen Sprachen mit lateinischem Alphabet kommt unter Windows die Codierung Windows-1252 oder CP-1252 zum Einsatz.

Befindet sich in der Textdatei eine BOM, sollte man diese vor dem eigentlichen Import entfernen. Die BOM dient als Kennung zur Definition von Byte-Reihenfolge und Codierungsform.

Einen ersten Schritt für eine direkte automatisierte Übernahme bildet der von Michael Hunger implementierte Neo4j-RDBMS-Importer (ursprünglich Relational to Neo4j Importer genannt); dieser befindet sich seit etwas mehr als einem Jahr in der Erprobungsphase.

Einsatz mit normalisierten relationalen Datenbanken

Als Werkzeug steht der Neo4j-RDBMS-Importer bereits für den Einsatz mit normalisierten relationalen Datenbanken zur Verfügung, da diese die Transformationsregeln zur Überführung erfüllen.

Das SQL-Import-Werkzeug entscheidet anhand dieser Regeln, wann aus einer relationalen Tabelle ein Knoten oder eine Beziehung im Graphen wird. Zudem überführen sie Spalten in Properties der Knoten/Beziehungen des Graphen und konvertieren eventuell Datenwerte in Neo4j-spezifische Datentypen.

Michael Hunger zeigte die Praxistauglichkeit der Implementierung für die aus der Neo4j-Dokumentation bekannte Northwind-Datenbank sowie für zwei weitere aus der MySQL-Dokumentation bekannte Demo-Datenbanken (Sakila und Employees).

Leider findet man in der Praxis oft nicht derart einfache Datenstrukturen wie bei den Demo-Datenbanken vor; vielfach mangelt es auch an einer ausreichenden Datenqualität. So stößt man in der Praxis zum Beispiel häufig auf einen historisch gewachsenen Missbrauch von Feldern. Für eine vorausgehende, grundlegende Erkundung des Schemas einer relationalen Datenbank eignen sich Werkzeuge wie SchemaCrawler oder SchemaSpy.

Erschwerend kommt hinzu, dass die Abbildung von Beziehungen in relationalen Datenbanken nicht immer durch einfache Übernahme von Primär- als Fremdschlüssel erfolgt. Derartige Konstellationen können von keinem Automatismus unmittelbar berücksichtigt werden.

Auch bleiben Fragen offen, ob und wie besondere Mechanismen eines relationalen DBMS wie zusammengesetzte Indizes, fortgeschrittene Constraints, Trigger, Stored Procedures und Ähnliches zu behandeln sind.

Insgesamt sollten jedoch für den Neo4j-RDMBS-Importer aufgrund seiner Bedeutung für die Zukunft weitere Anstrengungen unternommen werden. So ermöglicht ein erfolgreicher Direktimport beispielsweise einen Parallelbetrieb von SQL- und Graph-Datenbank für eine wesentlich schnellere Informationsanalyse. ■

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

Links zum Thema

- Common Format and MIME Type for CSV Files
  http://tools.ietf.org/html/rfc4180
- 30 Best Free CSV Editor Software For Windows
  http://listoffreeware.com/list-of-best-free-csv-editor-software-for-windows
- The 4 Best CSV Files Freeware Editors
  www.softmazing.com/the-4-best-csv-filles-freeeware editor.
- Homepage der Online CSV Conversion Tools
  www.convertcsv.com
- Homepage von Mockaroo
  www.mockaroo.com
- Homepage von Papa Parse
  http://papaparse.com
- Homepage von CSVLint
  http://csvlint.io
- Homepage von CSV Validator 1.1
  http://digital-preservation.github.io/csv-validator
- Website Online Generate CSV Test Data
  www.convertcsv.com/generate-test-data.htm
- Homepage der csvkit-Toolsuite
  http://csvkit.readthedocs.org/en/540
- Neo4j (CSV) Batch Importer von Michael Hunger
  https://github.com/jexp/batch-import
- Neo4j-Shell-Tools von Michael Hunger für den Datenimport
  https://github.com/jexp/neo4j-shell-tools
- Blog-Post: Building an Import Tool for Relational to Neo4j over one Weekend
  https://github.com/jexp/neo4j-rdbms-import/blob/master/docs/relational_to_neo4j_import_tool.ado
- Relational to Neo4j Importer von Michael Hunger
  https://github.com/jexp/neo4j-rdbms-import#relational-to-neo4j-importer
- Dokumentation des Relational to Neo4j Importer
  https://github.com/jexp/neo4j-rdbms-import
- Neo4j – Northwind Example Model
  http://neo4j.com/developer/guide-sql-to-cypher/#_northwind_example_model
- MySQL – Employees Sample Database
  https://dev.mysql.com/doc/employee/en/sakila-structure.html
- MySQL – Sakila Sample Database
  https://dev.mysql.com/doc/sakila
- Homepage von SchemaCrawler
  http://sualeh.github.io/SchemaCrawler
- Homepage von SchemaSpy
  http://schemaspy.sourceforge.net

Downloaden, aufschlauen!

PAGE
36 Seiten
eDOSSIER 09.2014
shop.page-online.de/downloads

Typografie im Web

FAQs: Formate, Lizenzen, Stolperfallen: Antworten auf all die Fragen, die Sie schon immer einmal stellen wollten

Step by Step: Webfonts in einen Internetauftritt einbinden

Responsive Typography – Notwendigkeit oder Nonsense: Typo, die sich an jedes Endgerät anpasst

PAGE
28 Seiten
eDOSSIER 02.2014
shop.page-online.de/downloads

Projektmanagement in der Kreativbranche

Kreation, Budget, Timing und Technik stets im Griff

Agiles Projektmanagement: So finden Sie die richtige Methode für Ihr Projekt

Fallbeispiel: Scrum bei XING

PAGE
31 Seiten
eDOSSIER 03.2014
shop.page-online.de/downloads

Selbstständig in der Kreativbranche

Existenzgründung: Fallbeispiele, Fakten, Tipps für Freelancer und Agenturen

Positionierung, Businessplan, Versicherung, Akquise & Co

Der Sprung in die Selbstständigkeit ist geschafft – und dann?

Storytelling für Start-ups

PAGE
13 Seiten
eDOSSIER 12.2014
shop.page-online.de/downloads

Kreative Recherche: Inspiration durch Information

Die Strategien, Tools und Tricks von Werbern, Gestaltern, Journalisten und Trendforschern

Praxisbeispiele: Bureau Mirko Borsche, rethink, Jung von Matt/Spree

PAGE
33 Seiten
eDOSSIER 05.2014
shop.page-online.de/downloads

3-D: Druck und Scanning

Von Low- bis Hightech: 3-D-Printer und 3-D-Druckservices

3-D-Scanning: Von der Gratis-App über Low-end-Scanner bis Hightech-System

Spannende Anwendungen und Geschäftsideen

PAGE
19 Seiten
eDOSSIER 06.2014
shop.page-online.de/downloads

Kreative Konzepte visualisieren

Schnell und schlüssig komplexe Ideen greifbar machen

Techniken für Briefing, Kundenpräsentation und Usabilty-Überprüfung

Vom Scribble über haptische und 3-D-Modelle bis hin zum Screen-Mockup

PAGE
37 Seiten
eDOSSIER 08.2014
shop.page-online.de/downloads

Arduino: Grundlagen und Workshops

Werkzeuge und Bauteile für eine kreative Tinkerbox

Step-by-Step-Anleitungen:
– Der Arduino Art Pen
– Gesichtssteuerung mit FaceOSC
– Blickinteraktion mit haptischem Feedback
– NFC-Automat

PAGE
21 Seiten
eDOSSIER 07.2014
shop.page-online.de/downloads

Die Agentur der Zukunft

Fallbeispiele: Kleine Agenturen in Zeiten ausdifferenzierter 360-Grad-Kampagnen

Das können Design- und Werbeagenturen tun, um innovationsfähig zu bleiben

Netzgesellschaft, schrumpfende Märkte – was kommt auf Digitalgenturen zu?

PAGE
52 Seiten
eDOSSIER 10.2014
shop.page-online.de/downloads

Responsive Webdesign: Basics, Cases, Workshops

Worauf man bei der Gestaltung responsiver Websites achten muss

Tutorials: Adobe Edge Reflow, Axure 7, Zurb Foundation, SVG, CoffeeScript & Co

Fallbeispiele:
– Die Ferrari-Website von SinnerSchrader
– Die HTML5-Webapp von Lufthansa

PAGE
39 Seiten
eDOSSIER 11.2014
shop.page-online.de/downloads

Frei oder angestellt in der Kreativbranche

Welches Erwerbsmodell passt zu mir? – Praxisbeispiele & Entscheidungshilfen

Festangestellten-Gehälter, Honorarrichtlinien für Freelancer und Verhandlungstipps

Agentur adē – hallo Wirtschaft: Arbeitsbedingungen und Tätigkeitsfelder für Angestellte und Selbstständige

PAGE
31 Seiten
eDOSSIER 14.2014
shop.page-online.de/downloads

E-Shop-Systeme für Designer

Shopping-Plattformen, Mietshops, Open-Source-Lösungen für den leichten Einstieg in den E-Commerce

PAGE
15 Seiten
eDOSSIER 13.2014
shop.page-online.de/downloads

Online-Portfolios: Tools, Tipps & Strategien

Selbstdarstellung im Web: Dos & Don'ts für Agenturen und Freelancer

Instant-Portfolios – online und mobile: Praktische Tools für Kreative ohne Programmierkenntnisse

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

TEMPLATE-ENTWICKLUNG FÜR MAGENTO 2.0

Individuelles Design

Auch bei der Template-Entwicklung hat sich bei Magento 2.0 einiges getan.

Seit dem 17. November 2015 stellt das Magento-Entwicklerteam ihre neue Version zum Download bereit. Nachdem der erste Run auf die neue Version abgeebbt ist und absehbar war, dass noch einige Zeit bis zur Einsatzbereitschaft der deutschen Version vergehen wird, haben Magento-Betreiber ausreichend Zeit, sich mit den spezifischen Eigenheiten der neuen Version zu beschäftigen.

Neben einem runderneuerten Backend wurde vor allem viel Arbeit in die Verbesserung der Performance gesteckt. Die neue Magento-Version soll jetzt bis zu 10.000.000 Views und bis zu 90.000 Bestellungen pro Stunde im Katalog verarbeiten können und damit in diesem Bereich rund 25 Prozent schneller sein als die Vorgängerversion. Auch am am Template-Mechanismus wurde bei Magento 2 geschraubt.

Spätestens dann, wenn Sie Ihrem Magento-2.0-Shop ein individuelles Design verpassen wollen, müssen Sie sich mit der Theme-Entwicklung befassen. Dazu müssen Sie die grundlegenden Konzepte kennen, die Struktur von Themes und einiges mehr. Bei der Theme-Entwicklung spielt das Zend Framework eine wichtige Rolle, und Magento macht sich die MVC-Architektur als Design-Muster zunutze.

Grundlagen der Template-Entwicklung

Prinzipiell liegen die von Magento verwendeten Themes im Ordner */app/design/frontend/entwickler/*. Bei dem Standard-Theme Luma differiert das Verzeichnis. Dessen Komponenten finden Sie im Verzeichnis */vendor/magento/theme-frontend-luma/*. Im Standard-Theme einer Magento-2.0-Installation sind eine Vielzahl von Unterverzeichnissen enthalten, die mit *Magento_* beginnen. Dabei handelt es sich um herstellerspezifische Funktionen, die in diesem Beispiel von Magento selbst stammen (Bild 1).

In der Theme-Struktur existierten drei Hauptdateien, die das Theme-Verhalten bestimmen:

- *composer.json:* Diese Datei beschreibt die Abhängigkeiten und die Meta-Informationen.
- *registration.php:* Diese Datei sorgt für die Registrierung der Themes im Magento-System.
- *theme.xml:* Diese Datei deklariert das Theme im System und wird von Magento für die Erkennung des Themes verwendet.

Man kann die Dateien in der Theme-Struktur in zwei Gruppen einteilen: solche, die für die statische, und solche, die für die dynamische Darstellung zuständig sind.

Bei den Dateien für die statische Darstellung von Inhalten ist keine Verarbeitung durch den Server notwendig. Dynamische Inhalte werden vom Server generiert, bevor diese an den Client übermittelt werden. Statische Inhalte findet man üblicherweise in den folgenden Unterverzeichnissen des Theme-Ordners:

- */pub/static/frontend/<entwickler>/<theme>/<sprache>*
- *<theme_verzeichnis>/media/*
- *<theme_verzeichnis>/web*

Wenn Sie Magento 2.0 auf einem lokalen Testsystem installieren und einen ersten Blick auf das Frontend – also auf die Shop-Darstellung für die Kunden – werfen, begegnen Sie dem Luma-Theme (Bild 2). Dieses Theme nutzt Responsive Webdesign, um sich an die Darstellungsmöglichkeiten des zugreifenden Clients anzupassen. Dabei kommen insbesondere CSS3 Media Queries zum Einsatz.

Magento nutzt die Möglichkeit von CSS, die Darstellung eines Dokuments für verschiedene Ausgabemedien festzulegen. Die Zuordnung eines Stylesheets zu einem Medium erfolgt mit Hilfe von Media Queries. Dabei wird eine Liste von Kriterien abgearbeitet, die die Ausgabegeräte erfüllen müssen, damit ein Stylesheet zur Verarbeitung eingebunden wird. Das Ergebnis ist eine optimale Darstellung der Inhalte auf unterschiedlichen Endgeräten.

Magento User Interface Library

Das Luma-Theme basiert außerdem auf der Magento User Interface Library. Diese Bibliothek ist ein Werkzeugkasten, aus

> vendor > magento > theme-frontend-luma

Name	Änderungsdatum	Typ	Größe
theme	23.03.2016 12:35	XML-Dokument	1 KB
registration	23.03.2016 12:35	PHP-Datei	1 KB
LICENSE_AFL	23.03.2016 12:35	Textdokument	11 KB
LICENSE	23.03.2016 12:35	Textdokument	11 KB
composer.json	23.03.2016 12:35	JSON-Datei	1 KB
web	23.03.2016 12:35	Dateiordner	
media	23.03.2016 12:35	Dateiordner	
Magento_Wishlist	23.03.2016 12:35	Dateiordner	
Magento_Theme	23.03.2016 12:35	Dateiordner	
Magento_SendFriend	23.03.2016 12:35	Dateiordner	
Magento_Sales	23.03.2016 12:35	Dateiordner	
Magento_Rma	23.03.2016 12:35	Dateiordner	
Magento_Reward	23.03.2016 12:35	Dateiordner	
Magento_Review	23.03.2016 12:35	Dateiordner	
Magento_Paypal	23.03.2016 12:35	Dateiordner	
Magento_Newsletter	23.03.2016 12:35	Dateiordner	
Magento_MultipleWishlist	23.03.2016 12:35	Dateiordner	
Magento_Msrp	23.03.2016 12:35	Dateiordner	
Magento_LayeredNavigation	23.03.2016 12:35	Dateiordner	
Magento_Invitation	23.03.2016 12:35	Dateiordner	
Magento_GroupedProduct	23.03.2016 12:35	Dateiordner	
Magento_GiftWrapping	23.03.2016 12:35	Dateiordner	
Magento_GiftRegistry	23.03.2016 12:35	Dateiordner	
Magento_GiftMessage	23.03.2016 12:35	Dateiordner	
Magento_GiftCardAccount	23.03.2016 12:35	Dateiordner	

Ein Blick in das Verzeichnis des Luma-Themes (Bild 1)

dem Sie sich sozusagen beliebig bedienen können. Sie können beispielsweise die folgenden Elemente in Ihren Shop-Seiten verwenden: Aktions-Werkzeugleiste, Schaltflächen, Dropdown-Menüs, Formulare, Nachrichten, Paginierung, Bewertungen, Registerkarten, Tabellen, Tooltipps und vieles mehr.

Das Luma-Theme greift außerdem auf Funktionen eines weiteren Themes zurück: das Blank-Theme. Dessen Daten finden Sie im Unterordner */vendor/magento/theme-frontend-blank*. Das Blank-Theme ist eine Art Master-Theme, das bestimmte Eigenschaften an das Luma-Theme vererbt. Man spricht daher in diesem Zusammenhang auch von Theme-Vererbung.

Zurückgreifen auf den Code des Blank-Themes

Diese Technik vereinfacht es Entwicklern, eigene Themes zu entwickeln, weil man immer wieder auf die Funktionen und damit auf den Code des Blank-Themes zurückgreifen kann. Man kann das Blank-Theme auch als Fallback verstehen, das immer dann zum Einsatz kommt, wenn in einem anderen Theme bestimmte Funktionen nicht implementiert sind. Der Vorteil: Der Entwickler muss nur die Funktionen implementieren, die für die Anpassung notwendig sind.

Die Vererbung muss natürlich in dem neuen Kind-Theme (das Ausgangs-Theme bezeichnet man entsprechend als Parent-Theme) deklariert werden. Im Luma-Theme wird das Parent-Theme in der Datei *theme.xml* wie in Listing 1 deklariert.

Diese Art der Vererbung funktioniert ähnlich wie ein Überschreibungssystem. Beim Erstellen eines eigenen Themes greifen Sie dabei immer auf die Konfiguration des Parent-Themes zurück und überschreiben die Konfigurationen, denen Sie eine eigene Gestaltung oder Funktionalität zuweisen wollen.

Listing 1: Deklaration des Parent-Themes

```
<theme xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=
"urn:magento:framework:Config/etc/theme.xsd">
  <title>Magento Luma</title>
  <parent>Magento/blank</parent>
  <media>
    <preview_image>media/preview.jpg</preview_image>
  </media>
</theme>
```

Wenn Sie ein neues Theme in */app/design/frontend/<entwickler>/<theme_name>/* anlegen und das Blank-Theme als Parent-Theme deklarieren sowie die Datei *theme.xml* und *registration.php* verwenden, so können Sie auf die gesamte Funktionalität des Blank-Themes zurückgreifen, einschließlich dessen Layout- und Stil-Einstellungen. Um nun eine eigene CSS-Datei zu verwenden, kopieren Sie diese nach */<theme_ordner>/web/css*.

Damit nutzt das Theme dieses Stylesheet. Wenn Sie die Datei wieder entfernen, greift das Fallback-System, und Magento sucht im Ordner *<parent_theme_verzeichnis>/web/css/* nach einer Stilvorlage.

Theme-Zuweisung

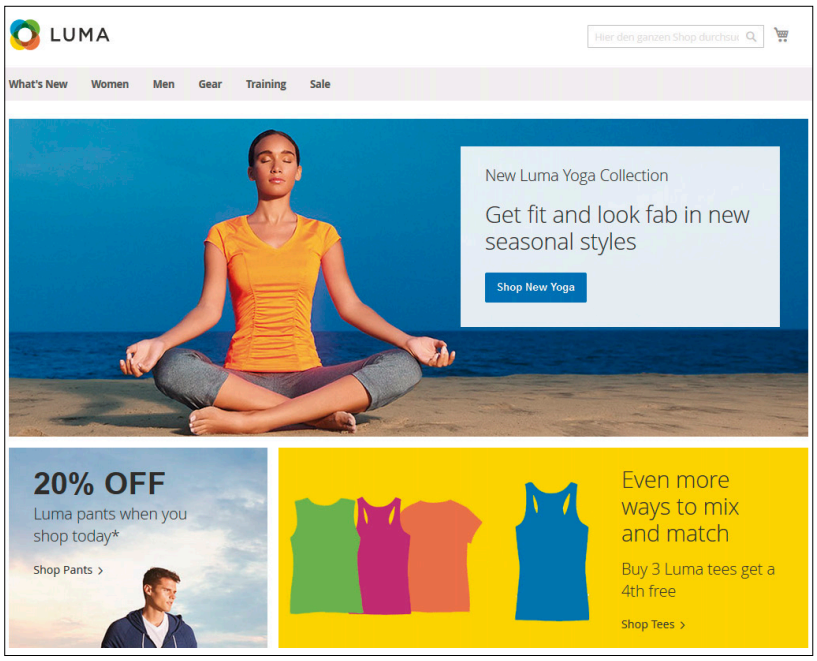
Sie können Ihren Magento-2.0-Shop um beliebige Themes erweitern. Wenn diese korrekt im System registriert sind, können Sie diese Shop-weit, aber auch für spezifische Seiten und Inhalte anwenden. Magento verfügt über ein durchaus praxistaugliches CMS-Modul, mit dem Sie Content-Blöcke und Seiten anlegen und verwalten. Die zugehörigen Funktionen sind über das Menü *Inhalt* verfügbar.

Im Untermenü *Seiten* verwalten Sie die statischen Inhalte Ihres Online-Shops, beispielsweise das Impressum, die Allgemeinen Geschäftsbedingungen (AGB), die Datenschutzerklärung et cetera.

In der Praxis kann es sinnvoll sein, diesen Inhalten ein anderes Design zuzuweisen. Dazu öffnen Sie die Seiteneinstellungen und wählen unter *Design* und *Benutzerdefiniertes Design* im Auswahlmenü *Benutzerdefinierte Themes* ein anderes Theme aus (Bild 3).

Benutzerdefinierte Variablen

Magento 2.0 erlaubt Entwicklern außerdem die Verwendung von benutzerdefinierten Variablen. Diese werden über das Menü *System, Andere Einstellungen* und *Benutzerdefinierte Variablen* angelegt. Benutzerdefinierte Variablen lassen sich ebenfalls für die Theme-Bearbeitung verwenden. ▶

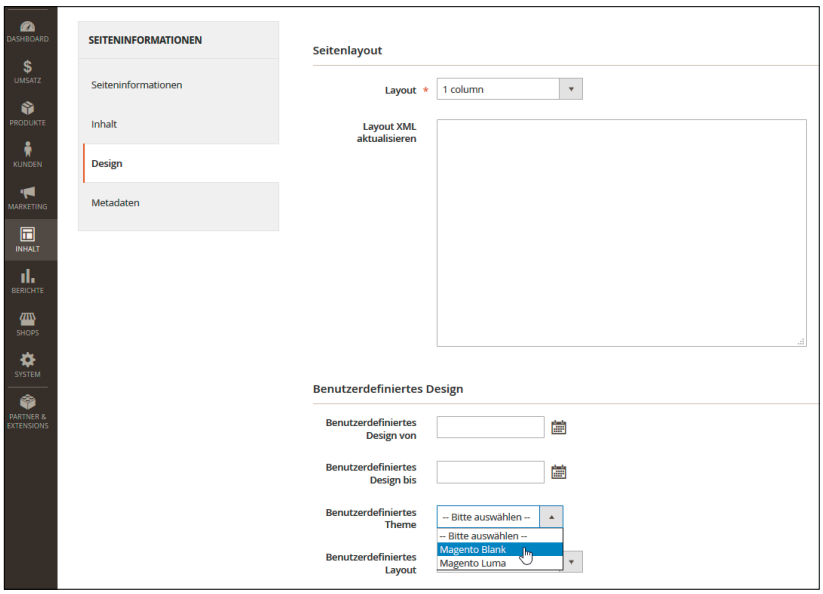


Das Standard-Theme von Magento 2.0 verwendet ein Responsive Design (Bild 2)

Mit dem grundlegenden Verständnis der Theme-Struktur und der Vererbung sind Sie für den nächsten Schritt gerüstet: den Einstieg in die Entwicklung eigener Themes für Magento 2.0. Was auf den ersten Blick komplizierter klingt, entpuppt sich in der Realität als deutlich einfacher.

Bevor Sie allerdings die ersten Schritte gehen, sollten Sie das Cache-Management deaktivieren, damit Updates unmittelbar umgesetzt werden können. Dazu öffnen Sie die Magento-Systemeinstellungen und wechseln mit *Tools* und *Cache-Management* zu den betreffenden Einstellungen. Wählen Sie alle Zwischenspeicher aus und deaktivieren Sie diese über das Auswahlmenü.

Alternativ können Sie auch zur Konsole greifen (eine weitere Neuerung von Magento 2.0 ist die Möglichkeit, das Shop-System auf der Konsole zu steuern). Dort führen Sie folgenden Befehl aus:



Die individuelle Zuweisung eines Themes zu CMS-Seiten (Bild 3)

```
php magento cache:disable
```

Auch dieser Befehl deaktiviert alle Zwischenspeicher des Magento-2.0-Systems. Dann erzeugen wir die grundlegende Theme-Struktur. In diesem Beispiel erzeugen wir das Verzeichnis *Beispiel (/app/design/frontend/Beispiel)*.

Deklaration der Theme-Informationen

In diesem neuen Verzeichnis legen wir anschließend einen weiteren Unterordner mit der Bezeichnung *basic* an *(/app/design/frontend/Beispiel/basic)*. Der nächste Schritt dient der Deklaration der Theme-Information. Dazu öffnen Sie Ihren bevorzugten Editor und erzeugen in dem *basic*-Verzeichnis die Datei *theme.xml*. Verwenden Sie in der XML-Datei den Code aus Listing 2 und speichern Sie die Datei mit den eingetragenen XML-Informationen.

Diese einfache Deklaration sorgt dafür, dass Ihr Magento-System Ihr Theme als Standard-Theme verwendet. Im Code sind der Theme-Name, der Verweis auf das Eltern-Theme und die Referenz zur Theme-Vorschau hinterlegt. Da wir bislang noch keine Vorschau und daher auch kein Vorschaubild erzeugen konnten, ist diese Konfiguration in der Datei *theme.xml* auskommentiert. Das können wir später nachholen.

Als Nächstes erzeugen wir in unserem neuen *basic*-Verzeichnis die Datei *registration.php*, die für die Theme-Registrierung sorgt. Kopieren Sie hierzu den Code aus Listing 3 in die Registrierungsdatei. Dieser Code registriert das neue Theme im Magento-System und informiert dieses über die neue Theme-Struktur.

Konfiguration des Produktbildes

Der nächste Schritt dient der Konfiguration eines einfachen Produktbildes. Sie können in einem Theme die Bildeigenschaften definieren, die die Produktdarstellung bestimmen. Dazu erzeugen Sie die Datei *view.xml*.

Konkret verwenden Sie hier das ID-Attribut. Auch für diesen Schritt müssen Sie wieder ein neues Unterverzeichnis an-

Listing 2: Die Theme-Informationen

```
<theme xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance" xsi:n
NamespaceSchemaLocation=
"urn:magento:framework:Config/etc/theme.xsd">
  <title>Beispiel-Theme</title>
  <parent>Magento/blank</parent>
  <!-- <media>
    <preview_image>
      media/preview.jpg
    </preview_image>
  </media>-->
</theme>
```

Listing 3: Die Registrierungsdatei

```
<?php
/**
 * Copyright © 2016 Magento. All rights reserved.
 * See COPYING.txt for license details.
 */
\Magento\Framework\Component\
ComponentRegistrar::register(
  \Magento\Framework\Component\
  ComponentRegistrar::THEME,
  'frontend/Brain/basic'
  __DIR__
);
```

legen, dieses Mal mit der Bezeichnung *etc (/app/design/frontend/Beispiel/basic/etc)*.

Der Pfad zur Darstellungskonfiguration sieht demnach wie folgt aus: */app/design/frontend/Beispiel/basic/etc/view.xml*. Kopieren Sie folgenden Code in die *view.xml*:

```
<image id="category_page_grid" type="small_image">
  <width>300</width>
  <height>300</height>
</image>
```

Die ID und die Typendeklaration bestimmen damit die Eigenschaften der kleinen Bilddarstellung.

Wenn Sie nun in diesem Verzeichnis auch statische Inhalte ablegen wollen, legen Sie das Unterverzeichnis *web* an. Das sollte außerdem die folgende Struktur erhalten:

```
/css
/source
/fonts
/images
/js
```

Ihrem Theme fehlt noch ein Logo. Magento 2.0 geht standardmäßig davon aus, dass das Logo die Dateibezeichnung *logo.svg* besitzt und im Verzeichnis */web/images/logo.svg* liegt. Wenn Sie eine andere Bezeichnung und ein anderes Format verwenden wollen, müssen Sie das entsprechend deklarieren. Eine solche Konfiguration überschreibt das Fallback-System.

Unser Logo soll das Dateiformat PNG besitzen und *logo.png* heißen. Für das Logo legen Sie das Verzeichnis */app/design/frontend/Beispiel/basic/Magento_Theme/web/images* an. Erzeugen Sie als Nächstes die Datei *default.xml* in dem Layout-Verzeichnis und kopieren Sie den Code aus Listing 4 in die XML-Datei.

Die Deklaration verwendet drei unterschiedliche Argumente zur Verwaltung der Logo-Attribute: *filename*, *width* und *height*. Wichtig ist an dieser Stelle, dass Sie die Attribute *ihre_logo_breite* und *ihre_logo_hoehe* durch die korrekten Werte ersetzen.

Damit ergibt sich bereits eine recht komplexe Struktur des neuen Themes. Sie müssen es nur noch anwenden. Dazu öffnen Sie die Menüfolge *Shops, Einstellungen, Konfiguration, Allgemein, Design*. Unter *Design Theme* können Sie über das Auswahlmenü das neu angelegte Theme auswählen.

Listing 4: Code für XML-Datei

```
<page xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
xsi:n
NamespaceSchemaLocation=
"urn:magento:framework:View/Layout/etc/
page_configuration.xsd">

<body>

<referenceBlock name="logo">
<arguments>
  <argument name="logo_datei" xsi:type="string">
    Magento_Theme/images/logo.png
  </argument>
  <argument name="logo_breite" xsi:type="number">
    ihre_logo_breite
  </argument>
  <argument name="logo_hoehe" xsi:type="number">
    ihre_logo_hoehe
  </argument>
</arguments>
</referenceBlock>

</body>
</page>
```

Links zum Thema

- Offizielle Website vom Magento
  https://magento.com
- Magento Connect
  https://www.magentocommerce.com/magento-connect
- Magento 2 Dokumentation für Entwickler
  http://devdocs.magento.com
- Magento 2 auf GitHub
  https://github.com/magento/magento2
- Offizielle Dokumentation
  https://magento.com/help/documentation
- Weiterführende Informationen und Links zu Magento 2
  https://firebearstudio.com/blog/magento-2-uberblick-funktionen-reviews-tutorials-demo-marz-2015.html

Fazit

Magento war schon immer ein agenturfreundliches System, weil Anpassungen am Basissystem, und hier insbesondere an Themes, nicht immer trivial sind. Dank der Vererbungstechnik dürfte sich die Theme-Entwicklung in Zukunft vereinfachen. ■

Holger Reibold
arbeitet seit über 20 Jahren als IT-Journalist und Autor mit den Schwerpunkten Internet, Open Source und Webtechnologien. Außerdem arbeitet er als E-Commerce-Manager.
Er bloggt unter
www.reibold.de

Bild: Shutterstock / ProStockStudio

OPENSTACK: EIN FRAMEWORK FÜR DIE CLOUD

OpenStack und seine Dienste

Openstack ist eine Plattform für den Aufbau von virtualisierten Systemen und Clouds.

Alle heute genutzten IT-Dienste brauchen immer drei zentrale Ressourcen: die CPU mit Arbeitsspeicher (Compute-Power) um die Programme abzuarbeiten, Netzwerke zur Kommunikation mit anderen Systemkomponenten und Benutzern, und Massenspeicher (Festplatten) zur Ablage von Daten oder Programmen. Hinzu kommen natürlich noch all die weiteren IT-Baugruppen, auf die hier nicht näher eingegangen werden soll.

Durch die Betriebssysteme wird die Rechenleistung – die Compute-Power – den Anwendungen bereitgestellt. Betriebssysteme und Server werden seit Jahren erfolgreich virtualisiert. Und auch für die Netzwerke findet sich mittlerweile ein breites Spektrum an virtuellen Techniken. Die Virtualisierung des Speichers allerdings hinkte bislang dieser Entwicklung hinterher. Das soll sich mit OpenStack ändern (http://docs.openstack.org). OpenStack setzt sich aus einer Gruppe von Bausteinen zusammen, die sich unter anderem um die Virtualisierung sowie die Bereitstellung von Storage kümmern (Bild 1).

OpenStack ist, verglichen mit den anderen Virtualisierungsebenen wie Compute oder Netzen, recht jung, aber nicht minder komplex. In diesem Beitrag gehen wir vor allem auf den Aspekt des Speichers ein.

Das Ziel von OpenStack

Das Ziel von OpenStack und den bestehenden Virtualisierungstechniken ist, eine höhere Dynamik zu erreichen und die IT-Systeme besser für die Cloud-Nutzung fit zu machen. Es soll damit möglich sein, die gesamte Infrastruktur, die für den Betrieb von Anwendungen benötigt wird, als Dienst bereitzustellen. Hierbei wird oftmals auch von Infrastructure as a Service gesprochen.

OpenStack entwickelte sich aus der Zusammenarbeit des US-Webhoster Rackspace und der US-Raumfahrtbehörde NASA. Das Ziel der NASA war eine dynamische Bereitstellung von Rechenleistung, just wie sie heute durch Cloud-Angebote abgedeckt wird. Die NASA wollte durch das Projekt, das damals als Nebula bezeichnet wurde, Rechenleistung an

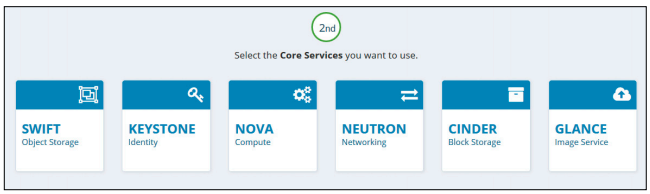
zentraler Stelle bündeln und den einzelnen Abteilungen nach Bedarf zuzuweisen. Anforderungen dieser Art finden sich heute in allen Cloud-Lösungen. Die NASA ist aktuell nicht mehr involviert. Dafür aber haben sich andere Unternehmen, wie etwa HP, IBM oder Intel, angeschlossen und treiben OpenStack nun weiter.

Die etablierten Linux-Anbieter haben OpenStack-Anwendungen im Portfolio. Auf der Webseite von OpenStack findet sich die Software zum Download für die Linux-Derivate von SUSE, Redhat und Ubuntu. OpenStack ist modular und besteht aus vielerlei Komponenten. OpenStack umfasst eine Reihe von Key Services (Bild 2):

- Nova: Die Compute-Power von OpenStack.
- Neutron: Hierin sind die Netzwerktechniken zusammengefasst.
- Keystone: Dient zur Identitätsverwaltung.
- Cinder: Stellt blockbasierten Speicher bereit.
- Glance: Umfasst die Image-Dienste von OpenStack.
- OpenStack Object Storage (Swift): Speicherverwaltung.

Virtualisierung und Clouds zielen vor allem auf eine bessere Ressourcenausnutzung und die dynamische Bereitstellung von Diensten ab. Cloud-Konzepte sind bei der heutigen IT-Nutzung nicht mehr wegzudenken. Für einen Zugriff von unterwegs auf die eigenen Daten oder aus dem Homeoffice sind sie unabdingbar. Unternehmen und Privatanwender nutzen daher diese neuen Techniken. Unabhängig von der konkreten Implementierung, ob als Private Cloud, Public Cloud oder Hybrid Cloud, ist die Nutzung kaum aufzuhalten. Und mit ihr entwickelt sich die Technik weiter fort. Dies wird auch in Zukunft so fortgeschrieben.

Der Grund für die sich ausbreitende Cloud-Nutzung zeigt sich dann, wenn man den Gebrauch und Nutzen von der Technik und den Hypes separiert. Die Ursachen für die expandierende Cloud-Nutzung liegen im Nutzerverhalten von Privatpersonen und Unternehmen mitsamt ihren Beschäftigten. Die Mitarbeiter der Unternehmen und private Nutzer verlangen nach immerwährender IT-Nutzung, auch von unterwegs oder zu Hause aus. Externe Mitarbeiter oder Subunternehmen müssen von außerhalb auf die Daten ihrer Geschäftspartner zugreifen können.



Dies sind die Kernfunktionen (Core Services) einer OpenStack-Implementierung (Bild 2)

Das könnten sie prinzipiell auch ohne Cloud, aber durch die Cloud-Angebote wird vieles einfacher oder erst möglich, wie wir noch zeigen werden. Möglich gemacht hat die Cloud-Nutzung wiederum die permanent fortschreitende Miniaturisierung der IT-Geräte. Ein Desktop-PC aus den 90er Jahren des vergangenen Jahrhunderts erlaubt nur eine stationäre Bedienung und Nutzung. Ein Notebook hingegen konnte man mitnehmen, das Gewicht aber schränkte den Nutzen ein. Die heutigen Smartphones und Tablets hingegen sind leichter und kompakter und liefern oftmals eine vergleichbare – wenn nicht gar bessere – CPU- und Grafikleistung.

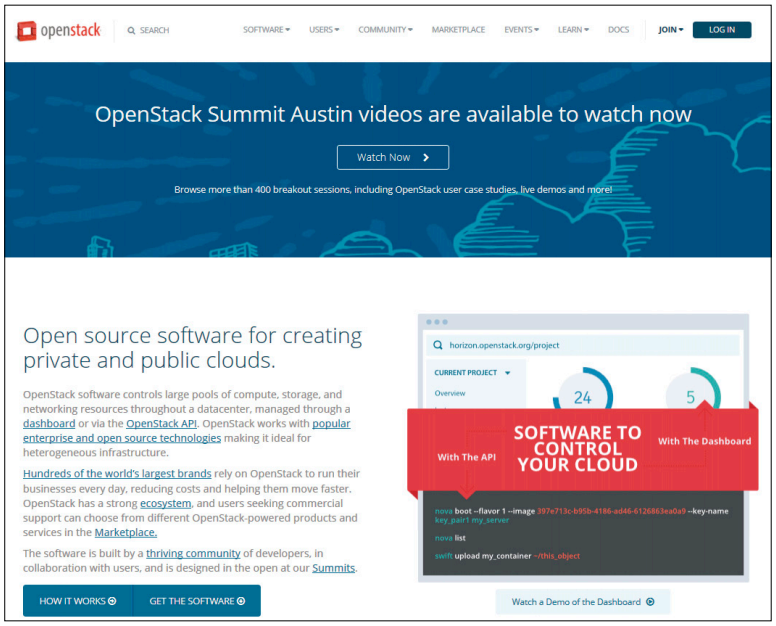
An der Cloud führt kein Weg vorbei

Die fortschreitende Miniaturisierung und die technischen Entwicklungen gelten aber nicht für alle Technologiebereiche gleichermaßen. Die Leistung der CPU, der Grafik, die Menge des Arbeitsspeichers, das Display, die Platten-Kapazitäten und die Netzanbindung wurden in den vergangen Jahren vervielfacht. Dennoch hinkt eine Ressource aller IT-Nutzung immer noch hinterher: die des externen Speichers, für den bis dato meist Festplatten eingesetzt werden.

Zwar sind Plattenkapazitäten von 2 und mehr TByte heute ohne große Kosten im 3,5-Zollformat verfügbar, aber für die Nutzung in einem Smartphone oder Tablet scheiden diese Platten gänzlich aus. SSDs sind zwar kompakter, mit 2,5 Zoll aber immer noch zu groß und schwer. Zwar kosten SSDs mit 500 Gigabyte heute weniger als 200 Euro, aber für den mobilen Einsatz sind sie zu schwer und benötigen wiederum zu viel Strom, was die Akkus vergrößern würde.

Eine Lösung wäre, wenn überhaupt, in Micro-SD-Karten zu sehen. Hierbei kosten derzeit 128 GByte circa 50 Euro. 1 TByte wäre damit oftmals teurer als die gesamte restliche Hardware des mobilen Geräts.

Somit bleibt für die Daten derzeit nur die Möglichkeit, diese auf anderen Systemen – eben in der Cloud – abzulegen. Zu den bekanntesten An- ▶



Auf der Webseite der OpenStack-Organisation findet sich ein Füllhorn an Software, Dokumentation und Hilfen (Bild 1)

bietern zählen wohl Google Drive, Microsoft OneDrive oder Dropbox. Daneben gibt es auch Reihe von weniger populären Angeboten wie etwa von der Telekom, Hidrive Free, Web.de oder MyTuxedo.

Just dieser Tage erschien eine Meldung zu Dropbox: Demnach speichern und bearbeiten weltweit 500 Millionen Nutzer mit Hilfe dieser Plattform gemeinsam Dateien und Dokumente. Das globale Wachstum macht deutlich, wie eng vernetzt Nutzer miteinander arbeiten. Seit der Gründung von Dropbox in 2007 haben sie 3,3 Milliarden Verbindungen aufgebaut, indem sie Dokumente miteinander teilen. Allein im vergangenen Jahr sei die Zahl dieser Verbindungen weltweit um 51 Prozent gestiegen. 44 Prozent der neuen Accounts sind entstanden, indem bestehende Nutzer ihre Freunde, Familie oder Kollegen zur Zusammenarbeit auf Dropbox einladen.

Cloud und Virtualisierung

Die Cloud aber nur als zentralen Datenspeicher zu sehen greift zu kurz. Die Cloud und deren Techniken sind vielmehr die Ergebnisse der fortschreitenden Virtualisierung jeglicher IT-Dienste. CloudStack ist eine Technik für den Aufbau und Betrieb von Cloud-Szenarien. Was sich hinter dieser Aussage verbirgt, wollen wir in den folgenden Zeilen erläutern. Cloud Computing stellt die neueste Variante der Bereitstellung von IT-Dienstleistungen dar. Dabei werden die Applikationen nicht mehr fest an einen Server gebunden. Stattdessen erfolgt die Verknüpfung der Applikationen nach Bedarf mit freien IT-Ressourcen.

Diese Trennung der Dienste von den Ressourcen ist die technische Voraussetzung für das Cloud Computing. Mit der Cloud gehen neue Verwaltungsprozesse und -abläufe einher. Dies verlangt nach neuen Toolsets, denn mit den bestehenden Systemverwaltungswerkzeugen ist die Dynamik, die die Cloud-Modelle versprechen, nicht zu machen. Zu den wichtigsten Funktionsblöcken beim Aufbau einer Cloud gehören die Poolbildung der Ressourcen, die Verwaltung und Bereitstellung von Templates und Software-Katalogen, die automatische Provisionierung der Dienste auf die virtuellen Server, die Kapazitätsplanung, die Verwaltung und Überwachung von SLAs, eine aufwandsgerechte Verrechnung der Dienste mit Cashback und die Abrechnung der verbrauchten Ressourcen (Bild 3).

Die Grundlagen der Cloud

Die Grundlagen jedes Cloud-Betriebs stellen heute virtualisierte Systeme dar. Dies gilt, weil die Cloud-Nutzer, im Gegensatz zu den Nutzern der traditionellen Inhouse-IT, im Voraus nicht bestimmbar sind. Beim traditionellen IT-Nutzer im Haus handelt es sich meist um die Beschäftigten des Unternehmens. Deren Anzahl und Arbeitszeiten sind im Voraus bekannt. Dasselbe gilt für die Menge der anfallenden Arbeiten und Batch-Jobs.

Der Cloud-Nutzer dagegen kann überall auf dem Erdball seine Dienste anfordern. Die Zahl der Nutzer und deren Nutzungsverhalten sind völlig unbekannt. Daher müssen die Cloud-Anbieter zwangsläufig mit einem Höchstmaß an Dynamik agieren. Diese kann eben nur durch die Virtualisierung erreicht werden.

Die Virtualisierung muss sich daher auf alle Ebenen erstrecken. Sie reicht von der Virtualisierung der Rechenleistung bis hin zu den Netzwerken, den benötigten Anwendungen und vollständigen Systemumgebungen. Hinzu kommt die Virtualisierung der Speichersysteme. Die traditionellen IT-Szenarien waren dabei klar getrennt. Um Rechenleistung für die Anwendungen zu virtualisieren – hierbei spricht man oftmals auch von den Compute-Diensten –, wurden spezielle Server eingesetzt.

Diese können natürlich – durch die Techniken der Servervirtualisierung – virtuell sein. Ähnlich verhält es sich mit den Desktops. In der Desktop-Virtualisierung werden diese Benutzergeräte virtuell abgebildet. Virtualisierte Netze entstehen durch die dynamische Bereitstellung von Netzwerkstrecken.

Neue Verwaltungswerkzeuge für die Cloud

Virtuelle Systeme und Cloud-Szenarien bringen neue Anforderungen an die Verwaltung. Um sie zu verwalten und zu überwachen, taugen die bestehenden Konzepte und Tools der Systemverwaltung nur bedingt. Traditionelle Management-Tools waren auf physische Systeme ausgerichtet. Die Virtualisierung der IT brachte neue Anforderungen.

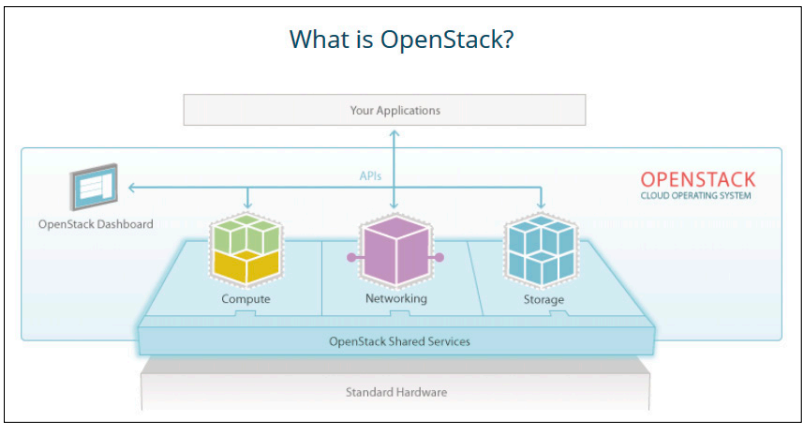
Dem begegneten die Anbieter durch passendende Verwaltungs-Tools für virtuelle Strukturen. Diese waren aber oftmals an die eigenen Hypervisoren und Verwaltungs-Tools angelehnt. Mit der Hinwendung zur Cloud kommen erneut neue Aspekte ins Spiel. Die bestehenden Verwaltungs-Tools bieten oftmals nicht den benötigten Grad an Automatisierung und weisen kaum den geschäftlichem Einblick auf, der von der IT benötigt wird, um mit den sich ändernden Geschäftsanforderungen Schritt halten zu können.

Im Gegensatz zu vielen traditionellen herkömmlichen Management-Tools sind die Verwaltungsmöglichkeiten von OpenStack an der Cloud ausgerichtet. Die erste Version als umfassendes System wurde 2010 veröffentlicht. Mittlerweile hat sich das System weiter gefestigt (Bild 4).

Die Hersteller bieten mittlerweile ein reichhaltiges Set an Tools und Hilfen in ihren Verwaltungssystemen (Bild 3)

OpenStack setzt dabei auf eine mehrstufige Architektur. Dazu gehören mehrere Software-Ebenen für den Betrieb virtueller Systeme:

- Nova: In Nova wird die Compute-Power von OpenStack eingruppiert. Dies sind virtualisierte Rechnersysteme. In der Übersicht von OpenStack werden dabei unter anderem die Begriffe »spawning, scheduling und decomissioning« erwähnt. »Spawning« ist den Linux-Anwendern nicht fremd. Es steht für das Erzeugen von Prozessen; im Kontext von OpenStack sind damit aber virtuelle Systemumgebungen gemeint. In der traditionellen Virtualisierung würde man wohl eher von Provisionierung sprechen. »Scheduling« ist die laufende Prozessverwaltung und CPU-Zuweisung an virtuellen Maschinen. Und beim »decomissioning« werden die virtuellen Maschinen wieder aufgelöst.
- Neutron: Hierin sind die Netzwerktechniken der virtualisierten Systeme in der Cloud zusammengefasst. Wenn eine Anwendung »on Demand«, also auf Anforderung eines Nutzers, aktiviert wird, so muss diese über virtuelle Netzwerkkanäle mit ihren Ressourcen und den Anwendern verknüpft werden können. Auch diese finden sich heute bei den bestehenden Virtualisierungsansätze in vergleichbarer Form.
- Keystone adressiert die Aspekte der Identitätsverwaltung und des Zugriffs auf die Objekte von OpenStack. Damit wird sichergestellt, dass nur autorisierte Benutzer Zugang zu den Ressourcen erhalten.
- Cinder stellt blockbasierten Speicher für Cloud-Dienste bereit.
- Glance umfasst die Image-Dienste von OpenStack. In diesen Images werden virtuelle Maschinen hinterlegt. Auch diese Technik ist aus der traditionellen Virtualisierung bekannt. Durch Automatismen werden IT-Dienste dynamisch bereitgestellt. Anstelle einer manuellen Provisionierung von Diensten und Systemen werden bei Cloud-Umgebungen diese virtuellen Systemumgebungen selbstständig eingerichtet und wieder aufgelöst. Die Grundlage für die Automatisierung wird in Regeln und Policies und den Richtlinien festgelegt. Diese bestimmen die Vorgehensweise für die Provisionierung. Automatismen verlangen nach klaren Vorgaben und Standards. Diese werden durch wiederverwendbarer Komponenten und Entwürfe gebildet. Die Verwaltungssoftware kümmert sich aber nicht nur um die Inbetriebnahme der IT-Dienste, sondern überwacht deren gesamten Lebenszyklus.
- OpenStack Object Storage (Swift) umfasst alle Aspekte bei der Speicherverwaltung. Auf die Speicherverwaltung werden wir im Folgenden noch vertieft eingehen.



Die Architektur einer OpenStack-Anwendung ist jener der Virtualisierung vergleichbar (Bild 4)

Für eine vollständige Dynamisierung der IT, wie sie für Cloud-Strukturen benötigt wird, bleibt nur noch die Virtualisierung der Speichersysteme. Dies erfolgt durch die Speichervirtualisierung. Auch sie ist nicht neu und wird bereits seit mehreren Jahren erfolgreich umgesetzt. Dennoch hinkte die Virtualisierung des Speichers allen anderen Disziplinen hinterher, dies gilt zumindest dann, wenn man etwas mehr Ansprüche an die Virtualisierung stellt.

Zwar werden bereits seit vielen Jahren Speicherkapazitäten auch dynamisch verwaltet, doch für die Ansprüche der Virtualisierung und der Clouds reichte dies meist nicht aus. Um auch den Speicher für die Cloud und Virtualisierung passend zu machen, mussten also neue Speicherverwaltungssysteme her. Den Anfang machten einfache Verwaltungshilfen, die man zwischen die anfordernden Rechnersysteme (die Compute-Ressource) und die Platten packte. Bereits die frühen Speichersysteme waren in der Lage, ihren Plattenplatz mehreren Nutzern (den Serversystemen) parallel zur Verfügung zu stellen. Dies wurde vor allem deswegen gemacht, um den Speicher effizienter nutzen zu können. Der Speicher einer im Server eingebauten Festplatte ist nur schwer für andere Systeme zugänglich.

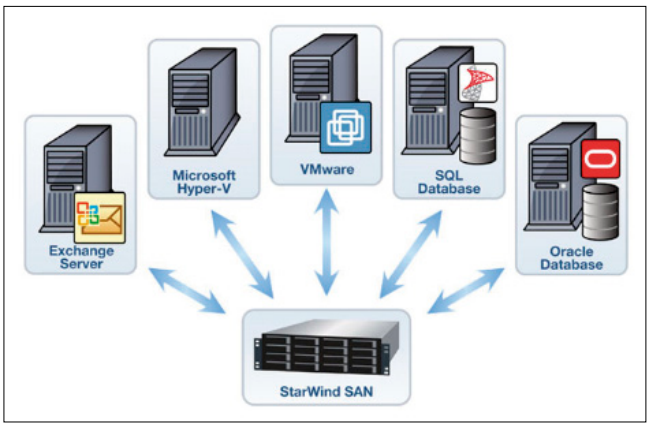
Konventionelle Speicher

In der Vergangenheit beherrschte konventioneller Speicher auf Block- oder File-Speichern die Speicherwelt. Dabei dominierten zwei unterschiedliche Verfahren. Sie teilten den Speichermarkt überschaubar in zwei überlappende Lager auf. Jeweils abhängig vom Anwendungsfall und der Kapazität konnten SAN-, NAS- und DAS-Segmente von einer Vielzahl von Herstellern mit ähnlichen Speichersystemen bedient werden.

Ein relativ unbekanntes Verfahren, um Speicher für andere bereitzustellen, stellt iSCSI dar. iSCSI oder Internet SCSI ist ein standardisiertes Protokoll zur blockbasierenden Datenübertragung via Ethernet und bietet damit ein Verfahren, das die Nutzung von iSCSI-Speicher über TCP/IP ermöglicht (SCSI over IP). OpenStack und auch NetApp (siehe unten) unterstützen iSCSI-Speicher. Das iSCSI-Protokoll kommuniziert mit iSCSI-Geräten. Es wird bevorzugt eingesetzt, um Servern den Zugriff auf Datenträger in Speichersystemen zu ermöglichen. Das Protokoll sendet auf Blockbasis und erstellt dazu eine virtuelle Punkt-zu-Punkt-Verbindung. So ist iSCSI schneller als die File-basierende Übertragung via CIFS ▶

oder NFS. Vereinfacht dargestellt enthält ein iSCSI-Netz mehrere Clients, Server und ein Speichersystem. Anders als Fibre Channel, das eine spezielle aktive und passive Infrastruktur benötigt, kann iSCSI über die bestehende IP- und Switch-Infrastruktur transportiert werden (Bild 5).

Der Client-PC, der innerhalb des Systems die Verbindung zu den gespeicherten Daten herstellen möchte, wird als iSCSI-Initiator bezeichnet. Zur Nutzung des iSCSI-Protokolls benötigt der Client oder Server einen software- oder hardwarebasierenden iSCSI-Initiator. Er übernimmt den Part, den

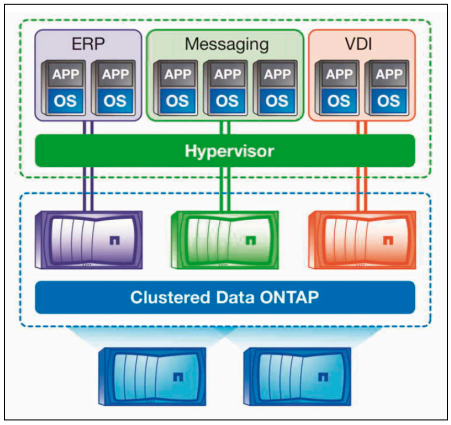


Über iSCSI lassen sich virtuelle Systeme mit externem Speicher verknüpfen (Bild 5)

bei einem PC normalerweise der SCSI-Bus-Adapter ausfüllt, nur dass anstelle der physischen Verkabelung der SCSI-Geräte die Befehle bei iSCSI über TCP/IP übermittelt werden.

Dennoch: Der schnellste Speicher wird immer der lokal angebunden Speicher sein. Wenn aber der Speicherbedarf starken Schwankungen unterliegt, ist die Situation völlig anders. Um einem Server zusätzlichen lokalen Speicher zu geben, musste der Server heruntergefahren und eine weitere Festplatte eingebaut werden. Dies ist schon bei internen IT-Strukturen mit viel Aufwand verbunden, für Cloud-Dienste ist es schlichtweg untragbar. Also hat man die Systeme für den maximalen Speicherbedarf ausgelegt. Dies aber hat zur Folge, dass sich der Plattenplatz bei stark schwankenden Speicherplatzanforderungen meist am oberen Limit orientierte. Dies wiederum führt zu einer geringen Effizienz des verwendeten Speichers, da er meist eben ungenutzt war.

Die Lösung war und sind auch heute noch zentrale Speichersysteme, wie sie beispielsweise von EMC, NetApp, HP oder IBM angeboten werden. Sie erlauben die dynamische Bereitstellung von zusätzlichem Speicherplatz. In der Sprache der Speicherhersteller spricht man dabei von LUN (Logical Unit Number). Eine LUN ist dynamisch allo-



Die Architektur der Manila Files Services von NetApp (Bild 6)

zierter Speicherplatz, ähnlich einer Partition. Aber im Unterschied zu den lokalen Partitionen von Festplatten liegt die LUN eben auf einem zentralen Speichersystem – und nicht lokal auf einer Platte, die im Server eingebaut ist.

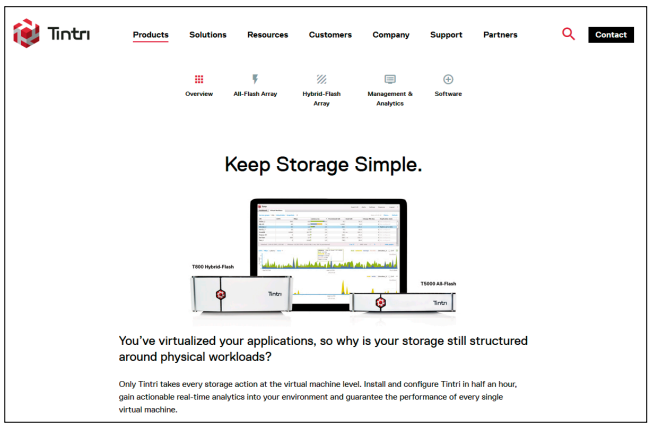
Der Administrator kann die LUN einem anfordernden Dienst auf Anfrage zuweisen. Damit wird der Speicher zentral verfügbar und kann je nach Bedarf einem Serversystem zugeteilt werden. Um die Dynamik zu erhöhen, kann die gesamte Verwaltung und Zuweisung aber auch durch eine Verwaltungssoftware erfolgen. Nun muss kein Administrator manuell eingreifen. Stattdessen übernimmt die Verwaltungsschicht die Erstellung und Zuweisung der LUN an den Serverdienst. Speicher, der in LUNs verwaltet wird, ist somit weitaus dynamischer und flexibler als fest eingebaute Platten.

Dennoch war das nur der Anfang auf dem Weg zu einem vollständig dynamischen Plattenplatz. Ursprünglich wurde der einer LUN zugewiesene Speicherplatz fest reserviert und war damit für keine weitere LUN verfügbar. Für Systeme mit kontinuierlich wachsendem Speicherbedarf aber ist es dennoch nur suboptimal. Wenn etwa einem Mailserver eine LUN von 2 TByte zugewiesen wurde und dieser den benötigten Platz der 2 TByte erst in sechs Monaten benötigt, dann wird dabei viel Speicherplatz, der zwar nun zentral verteilt werden kann, reserviert, obwohl er erst in Monaten benötigt wird. Um dem abzuhelfen, könnte man natürlich die LUN-Größe senken.

Effizienz der Speichernutzung

Wenn statt einmalig 2 TByte eben 5 x 400 GByte, zeitlich abgestuft, zugewiesen werden, so würde sich die Effizienz der Speichernutzung erhöhen. Gleichzeitig aber erfordert diese mehr Verwaltung und zerstückelt den verwalteten Speicher in viel mehr kleinere Segmente. Außerdem erfordert es mehr Zurückhaltung von Seiten der Administratoren, wirklich nur den Speicher anzufordern, der in naher Zeit benötigt wird. Um das Problem zu lösen, führte man die Technik des Thin Provisioning ein. Dabei wird durch den Administrator oder die Verwaltungssoftware der Speicher in größeren Einheiten angefordert, aber nur nach Bedarf zugewiesen. Dies passiert durch eine Verwaltungsschicht, die sich um Thin Provisioning kümmert.

Hierbei wird nicht der gesamte Speicher der LUN von Anbeginn an fest für diese LUN zugewiesen. Stattdessen erhält die LUN nur einen Bruchteil ihrer Größe. Der weitere Speicher wird nach Bedarf an die LUN übergeben. Der Vorteil dabei: Der physische Speicher muss für eine LUN nicht fest reserviert werden, sondern kann auch an andere LUNs übergeben werden. Natürlich darf die Summe aller den LUNS tatsächlich übergeben Speicherfragmente nie größer sein als die Summe des tatsächlich vorhandenen Speichers. Aber die Technik des Thin Provisioning erlaubt eine Mehrfachzuweisung,



Anders als bei traditionellen Arrays, welche die VMs nicht differenzieren und sie einfach durch den FIFO-E/A-Mischer schleusen, erhält bei Tintri jede VM ihren eigenen Kanal (Bild 7)

je nach Bedarf. Für Virtualisierung und Cloud ist die Technik des Thin Provisioning daher unabdingbar. So kann man beispielweise einer Anwendung, die zum Monatswechsel viel Speicher benötigt, diesen temporär zuweisen. Nach dem Monatsabschluss allerdings fällt dieser Speicher wieder in den Pool des Speichersystems zurück.

So hilfreich und notwendig Thin Provisioning und Speichersysteme sind, für Clouds und Virtualisierung sind sie dennoch nicht ausreichend. Hierfür werden noch weitere und feinere Abstufungen der Speicherzuweisung und Speichernutzung benötigt. Thin Provisioning adressiert nur eine Dimension der Speicherverwaltung: die der Kapazität oder Größe der zugewiesenen Speicherfragmente. Für vollständig virtualisierten Speicher muss aber weitaus mehr gelten. So wird beispielsweise die Auslastung des Speichersystems hinsichtlich seiner Übertragungswege (IO-Kapazität) durch Thin Provisioning nicht berührt. Auch die weitere Verwaltung oder etwa die Bereitstellung von Speicher in der Cloud muss gelöst werden. Hier wiederum setzt OpenStack an.

OpenStack Swift

Zu OpenStack Swift gehören alle diese Techniken für die Speicherverwaltung. Dazu zählen die Techniken für Clustering, Redundanz und Skalierbarkeit. OpenStack basiert auf einer verteilten Architektur und Verwaltung. OpenStack-Daten (Objekte) werden auf mehreren Speichersystemen abgelegt. Damit deckt das System die Aspekte der Redundanz und Ausfallsicherheit ab. Beim Ausfall eines Knotens greift das System automatisch auf die Datensicherung des zweiten Knoten zurück. Storage Cluster werden automatisch auf weitere Systeme repliziert. Wie erwähnt ist OpenStack relativ neu. Es wird aber bereits von einer Reihe von Produkten unterstützt. Im Folgenden zeigen wir zwei Beispiele mit Modellcharakter für die OpenStack-Nutzung.

NetApp zählt zu den bekannten und etablierten Speicherherstellern. Das Unternehmen hat unlängst den Manila File Share as a Service als produktionsreife OpenStack-Funktion freigegeben. Mit dem von NetApp begründeten Open-Source-Projekt Manila können Unternehmen ein Cloud-fähiges Rechenzentrum für geschäftskritische Anwendungen und Content-Management-Applikationen aufbauen. Anwender können damit auch Speicher- und Testumgebungen in Big Data-Größenordnung unterstützen, in denen Daten flexibel über unterschiedliche Clouds bewegt werden. Mit Manila soll außerdem ein freier Austausch zwischen öffentlichen und privaten Clouds möglich sein. NetApp ist Gründer des Manila-Projekts und Mitglied der OpenStack-Foundation.

Manila wurde als Baustein für OpenStack entwickelt. Es bietet einen automatisierten On-Demand-Service für die Bereitstellung von gemeinsamen und verteilten Dateisystemen. Dabei kommt ein offenes, standardisiertes API der OpenStack-Community zum Einsatz. Bisher konnten Shared Volumes nicht als Cloud-Dienst genutzt werden. Manila ermöglicht erstmals einen freien Austausch von Shared Volumes zwischen öffentlichen und privaten Clouds. Manila ist in der OpenStack Liberty Release verfügbar und ist daher auch darauf ausgerichtet, gemeinsam mit anderen Cloud-Stacks modular zum Einsatz zu kommen. Manila bietet darüber hinaus dank der Kooperation von NetApp mit der Open-Source-Community eine Auswahl an Backend-Optionen (Bild 6).

Tintri Operating System

Bei traditionellen Speichersystemen werden alle Speicheranforderungen durch einen gemeinsamen Kanal geschleust. Dies ist vergleichbar mit einer Warteschlange an der Kasse eines Supermarkts mit nur einer Kasse. Alle Einkäufe (IO-Anforderungen) müssen über die gleiche Kasse (den gleichen Kanal) abgewickelt werden. Dies hat Wartezeiten zur Folge.

Eine Priorisierung der Anforderungen nach Wichtigkeit oder Menge der angeforderten Daten findet nicht statt. Und so, wie man in Supermärkten seit Jahren mehrere Kassen für unterschiedliche Kundesegmente (Gold-Card-Kunde, Kunde mit nur fünf Artikeln) hat, lassen sich im Speichersystem die Anforderungen priorisieren. Dazu ist aber eine zusätzliche Verwaltungsebene notwendig. Diese sollte die virtuelle Maschine und deren Anforderungen berücksichtigen (Bild 7).

Traditionelle Speichersysteme machen keinen Unterschied zwischen den Prioritäten von virtuellen Verbrauchern. Die Anforderungen der Verbraucher werden meist der Reihe nach abgearbeitet. Eine Anwendung mit geringer Priorität, aber hoher Speichernutzung kann dabei Anwendungen mit hoher Priorität behindern. Tintri deckt mit seinem Operating System Aspekte der Verwaltung von virtualisiertem Speicher ab. Es unterstützt benutzerdefinierte Leistungsgarantien (IOPS) und -drosselung für Anwendungen. ■

Johann Baumeister
hat Informatik studiert und besitzt langjährige Erfahrung in der Entwicklung, Anwendung und dem Rollout von IT-Systemen. Außerdem ist er als Autor für zahlreiche IT-Publikationen tätig.

MIGRATION VON INHALTEN ZU WORDPRESS

Der große Umzug

Wie man den Umzug eines bestehenden Systems zum Content-Primus richtig macht.

Wenn Sie von einem anderen System auf WordPress wechseln möchten, soll das natürlich möglichst ohne Verluste vonstatten gehen. Sie möchten also Artikel, Bilder, Tags, Kategorien und User-Kommentare komplett mit übernehmen.

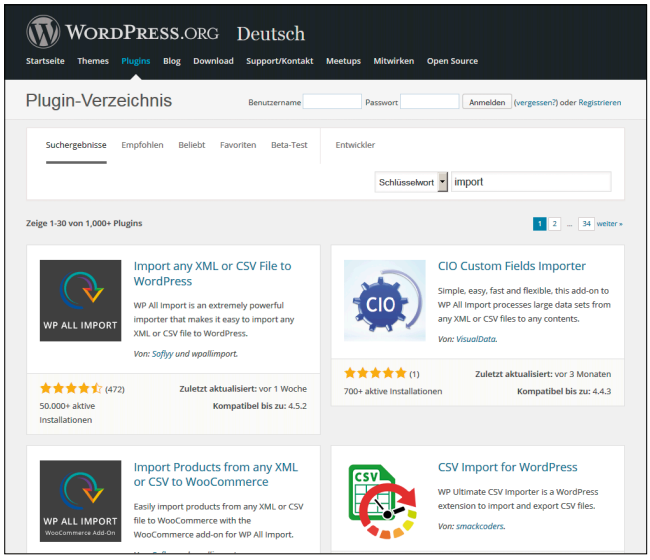
Am einfachsten ist der Umstieg, wenn Ihr bisheriges CMS zu den Systemen gehört, für die WordPress unter *Werkzeuge, Daten importieren* ein vorgefertigtes Importmodul zu bieten hat. Zusätzlich gibt es von WordPress noch eine Informationsseite, die für viele weitere CMS und Internetdienste einen automatisierten Importweg über Plug-ins oder Skripts aufzeigt (Bild 1).

Ist das nicht der Fall, etwa weil Sie mit einem weniger verbreiteten oder sogar selbst programmierten System arbeiten, so stellt sich die Frage, wie Ihr Content am besten in Richtung WordPress transportiert wird.

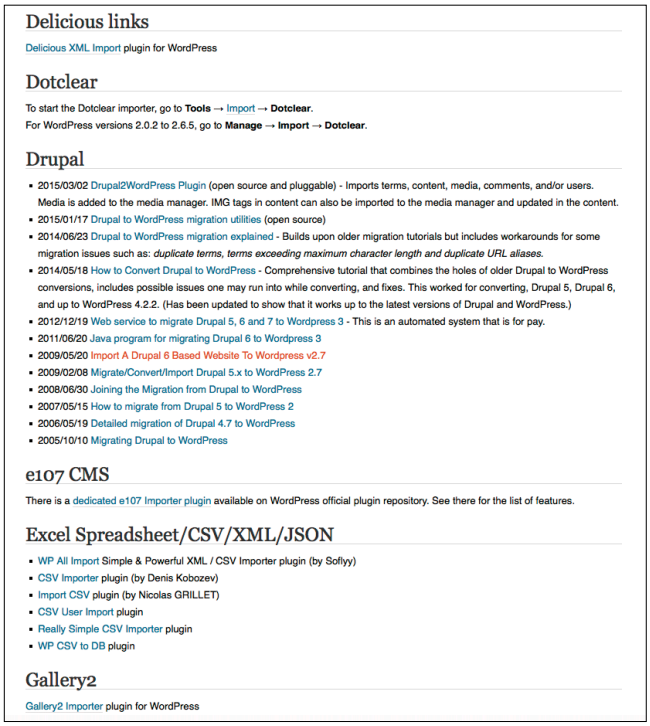
WordPress basiert aus MySQL

Da WordPress mit MySQL auf einem sehr verbreiteten Datenbanksystem aufsetzt und auch mit wenigen Tabellen auskommt, mag die direkte Bestückung der WordPress-Datenbank via SQL sinnvoll scheinen. Allerdings ist dieser Weg steinig.

Denn WordPress verfügt über ein Datenmodell, das extrem stark vernetzt und daher schwer nachzubauen ist. Außerdem könnte jedes Update von WordPress Details der Darstellung in der Datenbank ändern und somit Probleme erst lange nach dem Umzug ans Licht kommen lassen.



WordPress bietet eine ganze Reihe von Importwerkzeugen (Bild 1)



Delicious links

Delicious XML Import plugin for WordPress

Dotclear

To start the Dotclear importer, go to **Tools** → Import → **Dotclear**.
For WordPress versions 2.0.2 to 2.6.5, go to **Manage** → **Import** → **Dotclear**.

Drupal

- 2015/03/02 Drupal2WordPress Plugin (open source and pluggable) - Imports terms, content, media, comments, and/or users. Media is added to the media manager. IMG tags in content can also be imported to the media manager and updated in the content.
- 2015/01/17 Drupal to WordPress migration utilities (open source)
- 2014/06/23 Drupal to WordPress migration explained - Builds upon older migration tutorials but includes workarounds for some migration issues such as: *duplicate terms, terms exceeding maximum character length and duplicate URL aliases.*
- 2014/05/18 How to Convert Drupal to WordPress - Comprehensive tutorial that combines the holes of older Drupal to WordPress conversions, includes possible issues one may run into while converting, and fixes. This worked for converting, Drupal 5, Drupal 6, and up to WordPress 4.2.2. (Has been updated to show that it works up to the latest versions of Drupal and WordPress.)
- 2012/12/19 Web service to migrate Drupal 5, 6 and 7 to Wordpress 3 - This is an automated system that is for pay.
- 2011/06/20 Java program for migrating Drupal 6 to Wordpress 3
- 2009/05/20 Import A Drupal 6 Based Website To Wordpress v2.7
- 2009/02/08 Migrate/Convert/Import Drupal 5.x to WordPress 2.7
- 2008/06/30 Joining the Migration from Drupal to WordPress
- 2007/05/15 How to migrate from Drupal 5 to WordPress 2
- 2006/05/19 Detailed migration of Drupal 4.7 to WordPress
- 2005/10/10 Migrating Drupal to WordPress

e107 CMS

There is a dedicated e107 Importer plugin available on WordPress official plugin repository. See there for the list of features.

Excel Spreadsheet/CSV/XML/JSON

- WP All Import Simple & Powerful XML / CSV Importer plugin (by Soflyy)
- CSV Importer plugin (by Denis Kobozev)
- Import CSV plugin (by Nicolas GRILLET)
- CSV User Import plugin
- Really Simple CSV Importer plugin
- WP CSV to DB plugin

Gallery2

Gallery2 Importer plugin for WordPress

Die Website von WordPress führt einige Möglichkeiten auf, von bestehenden Systemen auf WordPress zu migrieren (Bild 2)

Eine Alternative wäre noch die Verwendung des REST-API. Diese Methode würde dafür sorgen, dass sich WordPress selbst um die Korrektheit und Integrität der Daten kümmert, also quasi als Business-Logik zwischen Ihrem Importmodul und den Daten agiert. Allerdings ist die Nutzung des API nicht ganz einfach, weil Sie sich mit vielen Details auseinandersetzen und dazu noch eine passende Client-Bibliothek für das Ansprechen des API finden müssen.

Gut, dass es einen einfacheren Weg gibt. Denn Sie können das Import-/Export-Format von WordPress (WXR) nutzen, um Ihren Content-Transfer mit vertretbarem Aufwand zu bewerkstelligen (Bild 2).

Dafür ist der XML-Export ausgelegt

Geschaffen wurde das System von WordPress, um eine Installation des CMS umziehen zu können. Dazu gehen Sie auf den Menüpunkt *Werkzeuge, Daten exportieren* und nutzen die Voreinstellung *Alles*. Nach dem Klick auf den Button bietet WordPress dann eine XML-Datei zum Download an.

Auf dem Zielsystem verwenden Sie dann *Werkzeuge, Daten importieren*, laden gegebenenfalls dabei noch das notwendige Import-Plug-in und geben die zuvor heruntergela-

dene Datei als Upload-File an. Der Importer analysiert daraufhin die erhaltene Datei und fragt, was er mit den gefundenen Beitragsautoren machen soll. Sie können sie neu anlegen lassen oder jeweils einem bestimmten vorhandenen User zuordnen.

Wichtig ist auf dieser Seite noch das unscheinbare Häkchen ganz unten bei *Dateianhänge herunterladen und importieren*. Nur wenn Sie das anklicken, bearbeitet WordPress Bilder und andere Medieninformationen und integriert sie auf dem Zielsystem.

Der Importvorgang fasst eventuell vorhandene Inhalte nicht an. Darum können Sie auf diesem Weg Inhalte häppchenweise in WordPress übernehmen.

So sieht das Export-Format WXR aus

Das Speicherformat von WordPress ist im Prinzip wie ein RSS-Feed aufgebaut: Die Klammerung *<rss><channel>...</channel></rss><channel>* enthält alle relevanten Informationen wie Autoren, Artikel oder Bilder auf derselben Ebene. Zuerst erscheinen einmalig die grundlegenden Daten der Website, wie der URL oder der Name. Darauf folgen Abschnitte für die Autoren, Kategorien und Tags. Erst im Anschluss folgen die *<item>*-Tags, die Beiträge und Seiten mit eventuellen Kommentaren beinhalten. Auch die Bilder sind als *<item>*-Abschnitte enthalten.

Der Inhalt von *<wp:post_type>* legt fest, ob es sich um einen Beitrag *(post)* oder eine feste Seite *(page)* handelt. Im Tag *<title>* ist die Headline eines Textbeitrags abgelegt, das Tag *<content:encoded>* nimmt seinen Inhalt auf und das Tag *<excerpt:encoded>* beinhaltet schließlich die Zusammenfassung, die auf der Titelseite oder in den Suchergebnissen erscheint.

Viele der weitereren Tags sind einigermaßen selbsterklärend, wie etwa *<wp:comment_status>* mit den Werten ▶

So klappt der Umzug via PHP

Wenn die Entscheidung feststeht, Ihre bisherigen Inhalte über das Speicherformat WXR zu WordPress zu übertragen, und Sie PHP als Werkzeug verwenden möchten, bleibt noch die Frage offen, über welche Methode dies geschehen soll.

Wenn es um die Erzeugung von XML geht, ist der sauberste Weg normalerweise ja immer die Verwendung einer XML-Library wie SimpleXML oder DOM. Allerdings ist die Struktur des verwendeten Dateiformats so einfach, dass man genauso gut mit normalen PHP-Strings arbeiten kann, ohne sich Probleme einzuhandeln.

Etwas praktikabler als die Nutzung von Strings ist ein Template-System wie zum Beispiel Twig, weil hier das XML nicht im PHP-Code steckt, sondern sauber getrennt als eigene Datei gehalten wird. Twig installieren Sie am besten über Composer mit *composer require twig/twig*.

Nehmen wir an, Sie haben bisher ein System, das die Beiträge für einen Blog in einer Datenbank hält. Die stecken alle in einer Tabelle *articles*, die Ihrerseits Spalten wie *head, body* oder *author* enthält.

Zuerst starten Sie eine Abfrage, die alle diese Artikel abruft und in ein PHP-Array ablegt:

```
$sql = "select head,body,author from articles";
$dbh = new PDO (...);
$stmt = $dbh->query($sql);
$articles = $stmt->fetchall(
                PDO::FETCH_ASSOC);
```

```
{% autoescape false %}
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0"
     xmlns:excerpt="http://wordpress.org/export/1.2/excerpt/"
     xmlns:content="http://purl.org/rss/1.0/modules/content/"
     xmlns:wfw="http://wellformedweb.org/CommentAPI/"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:wp="http://wordpress.org/export/1.2/"
>
    <channel>
        <title>Meine WP-Site</title>
        <language>de-DE</language>
        <wp:wxr_version>1.2</wp:wxr_version>
        {% for article in articles %}
        <item>
            <title><![CDATA[{{ article.head }}]]></title>
            <pubDate>Fri, 01 Jan 2016 12:00:00 +0000</pubDate>
            <dc:creator><![CDATA[{{ article.author }}]]></dc:creator>
            <description></description>
            <content:encoded><![CDATA[{{ article.body }}]]></content:encoded>
            <wp:post_date><![CDATA[2016-01-01 12:00:00]]></wp:post_date>
            <wp:comment_status><![CDATA[open]]></wp:comment_status>
            <wp:ping_status><![CDATA[closed]]></wp:ping_status>
            <wp:status><![CDATA[publish]]></wp:status>
            <wp:post_parent>0</wp:post_parent>
            <wp:menu_order>0</wp:menu_order>
            <wp:post_type><![CDATA[post]]></wp:post_type>
            <wp:post_password><![CDATA[]]></wp:post_password>
            <wp:is_sticky>0</wp:is_sticky>
            <category domain="post_tag" nicename="tag1"><![CDATA[Tag 1]]></category>
            <category domain="category" nicename="cat1"><![CDATA[Kategorie 1]]></category>
        </item>
        {% endfor %}
    </channel>
</rss>
{% endautoescape %}
```

Mit diesem Twig-Template können Sie aus PHP heraus einfache Beiträge erzeugen, die in WordPress importierbar sind

Nun öffnen Sie eine Instanz von Twig und übergeben die eben abgerufenen Daten dem Template *wp.xml.twig*:

```
$twig = new Twig_Environment(new
Twig_Loader_Filesystem(__DIR__));
$output = $twig->render('wp.xml.twig',
array('articles' => $articles))
```

Das Template-File *wp.xml.twig* sieht dann so aus wie im Listing. Durch das Statement *{% for article in articles %}* wird dort eine Schleife definiert, die alle Artikel behandelt. Der Schalter *autoescape false* in der ersten Zeile sorgt dafür, dass keine Sonderzeichen des HTML im Body oder in der Headline umgeformt werden, sondern wie gewünscht im Zielfeld landen. Das Ergebnis der Ersetzungsaktion via Twig steckt in der Variablen *$output*, die Sie entweder am Bildschirm ausgeben, in eine Datei speichern oder zum Download anbieten können.

open oder *closed*. Eine detaillierte Erklärung zur Bedeutung der anderen Felder finden Sie im Posting von Devtidbits.com, das in den Links zum Artikel genannt ist.

Nicht alle Felder müssen sein

Wenn Sie einmal zu Testzwecken eine mit Inhalt gefüllte WordPress-Installation exportieren und sich das XML ansehen, dann erschrecken Sie vielleicht über den Detailreichtum darin. Aber nicht alle der dort vorgefundenen Tags müssen Sie tatsächlich nachbilden und mit Inhalt füllen. Unnötig für Items sind beispielsweise die Einträge *<wp:post_date_gmt>*, *<link>* und *<guid>*, weil WordPress sich die Informationen von anderen Feldern holt.

Auch alle Tags, die eine eindeutige ID festlegen, wie *<wp:author_id>* oder *<wp:comment_id>*, können Sie weglassen. Denn WordPress vergibt dann selbstständig eine ID. Lediglich bei den Beiträgen ist das ID-Feld wichtig. Denn die Items für Bilder müssen als Referenz die Nummer des Beitrags enthalten, in dem sie erscheinen.

Ganz wichtig ist auch das Feld *<wp:post_name>*. Denn es bestimmt den URL des Beitrags in WordPress. Wenn Sie beispielsweise im Backend unter *Einstellungen* und *Permalinks* das benutzerdefinierte Format */%postname%.html* eingegeben haben und bei *<wp_post_name>* einem Beitrag den Namen *mein_cooler_text* geben, dann wird daraus der URL */mein_cooler_text.html*.

Wollen Sie also beim Umziehen den URLs eine bestimmte Form geben, um die Adressen aus dem alten System nachzubilden, dann müssen die Einstellungen für Permalinks und der im XML vergebene Post-Name genau zusammenpassen.

Auch die zusammengefasste Nennung der Tags und Kategorien im Kopfbereich vor den Items, wie sie der Export-Mechanismus anlegt, ist offensichtlich nicht unbedingt notwendig. Bei unseren Tests hat es ausgereicht, diese Informationen bei den jeweiligen Items aufzuführen.

Wenn Sie allerdings keine flache, sondern eine hierarchische Kategorisierung brauchen, dann müssen Sie zu Beginn des XML-Files die gewünschte Struktur aufbauen, also den Kategorien jeweils ihren Parent geben. Dazu nennen Sie den Nicename der übergeordneten Kategorie im Tag *<wp:category_parent>*. Daraufhin baut WordPress dann die Hierarchie korrekt auf.

So übertragen Sie Bilder und Videos auf die neue WordPress-Installation

Die eigentlichen Bild- oder Videoinhalte der Medien sind in der XML-Datei nicht eingeschlossen. Dort finden Sie lediglich Metadaten, wie den jeweiligen Speicherort oder die Ma-

Fehler finden und fixen

Natürlich wird Ihr erzeugtes XML nicht vom Stand weg genau das gewünschte Ergebnis produzieren – vielleicht wird es sogar vom WordPress-Importer verschmäht. Hier ein paar Tipps für den richtigen Umgang mit Problemen.

Ob das XML, das Sie mit Ihrem PHP-Code erzeugt haben, das richtige Ergebnis bringt, prüfen Sie am besten durch den Import in eine WordPress-Testinstallation. Ist dann noch Nacharbeit notwendig, ändern Sie Ihren Code und erzeugen ein neues XML. Für die Überprüfbarkeit ist es natürlich sinnvoll, das XML jeweils in ein leeres WordPress-System zu laden.

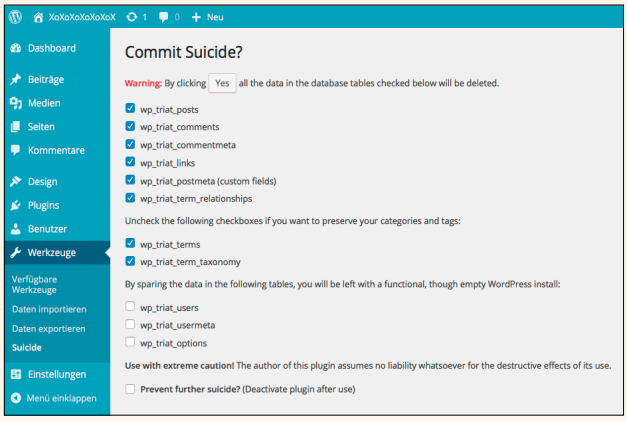
Diesen Total-Reset manuell zu bewirken erfordert einiges Herumgeklicke im Backend, das Sie sich sich mit dem Plug-in Suicide sparen können. Es entfernt standardmäßig alle Beiträge, Seiten, Kommentare, Tags und Kategorien. Sie könnten damit auch Ihre User-Daten löschen, was aber nicht empfehlenswert ist, weil Sie sich damit selbst den Zugriff entziehen.

Möchten Sie überprüfen, ob die Übernahme der Benutzer korrekt arbeitet, löschen Sie alle User außer Ihnen selbst im Backend: Wählen Sie alle eingetragenen Benutzer mittels der Checkbox links oben aus, dann wählen Sie sich selbst ab und verwenden die Aktion *Löschen*.

Es kann vorkommen, dass WP entweder die Annahme komplett verweigert oder Sie merken, dass etwas prinzipiell nicht stimmt. Ein typischer Fall ist dann, dass der Import statt der im XML angegebenen User lediglich einen Nutzer mit leerem Namen anzeigt.

Dann deutet einiges darauf hin, dass das XML nicht korrekt aufgebaut ist. Dessen Kontrolle ist manuell schwer zu bewerkstelligen. Besser ist die Verwendung eines XML-Validators. Dazu gibt es einige Kommandozeilen-Tools, oder Sie verwenden beispielsweise den Online-Validator von W3Schools, der in den Links zum Artikel genannt ist. Es gibt auch spezielle Validatoren für RSS. Die meckern dann aber auch über Abweichungen des von WP benutzten Formats, sodass Sie hier Mühe haben, Ihre eigenen Probleme von den Falschmeldungen zu trennen.

Es gibt auch Fälle, wo die Validatoren keine Probleme finden, WordPress aber trotzdem die Arbeit verweigert. So stellt das Programm beispielsweise oft dann die Arbeit ein, wenn Datumswerte nicht das korrekte Format haben.



Mit dem Plug-in Suicide löschen Sie mit einem Rutsch den gesamten Inhalt einer WP-Installation, was für Testzwecke praktisch ist

ße beispielsweise eines Bildes. Das ist auch gut so, denn sonst würden schnell die Grenzen des Formats gesprengt werden.

Beim Import werden diese Multimediadateien dann automatisch von ihrem alten Ablageplatz heruntergeladen und in die Mediathek des Zielsystems abgelegt. Das gilt aber nur für Medien, die im bisherigen System intern vorhanden waren. Medien, die bisher extern gespeichert waren, bleiben das auch weiterhin. Der Importer übernimmt dann also lediglich deren Verlinkung.

So erleichtert das XML-Format die Arbeitsweise

Damit die Übernahme von Bildern (und anderen Medien wie zum Beispiel Videos oder PDFs) klappt, müssen Sie für die Datei ein spezielles Item-Tag anlegen. Das braucht in *<link>* und *<wp:attachment_url>* die vollständige bisherige Adresse auf dem alten System. Zusätzlich muss im Tag *<wp:post_parent>* die ID des Posts genannt werden, auf den sich das Bild bezieht. Damit das möglich ist, müssen Sie also einen Zähler für Beiträge führen und bei dessen Item im Tag *<wp:post_id>* nennen und bei allen enthaltenen Bildern als Referenz im Tag *<wp:post_parent>* aufführen.

Das Schöne an der Verwendung des Speicherformats WXR ist, dass Sie damit beim Umzug auf Ihre WordPress-Installation inkrementell vorgehen können: Zuerst realisieren Sie zum Beispiel nur die Übernahme der Beiträge aus Ihrem alten System und überprüfen durch den Import der erzeugten Datei in WordPress, dass hier alle Elemente wie zum Beispiel die Headline, der Teaser oder die Formatierung des Inhalts korrekt funktionieren.

Klappt etwas bei der Übertragungsprozedur auf das Ziel-WordPress noch nicht so wie gewünscht, dann ändern Sie Ihrem Code zum Erzeugen des XML entsprechend und starten einen neuen Import.

Damit dieser Zyklus möglichst ohne großen Aufwand funktioniert, sollten Sie ein Tool zum Leeren Ihrer WP-Installation nutzen. ■

Abbruch wegen Platzangst

Je nach den Einstellungen Ihres WordPress-Hostings ist irgendwann die Grenze des maximalen Uploads einer WXR-Datei erreicht.

Das kann durch Erreichen der Upload-Größenlimits, einen Speicherüberlauf oder einen Timeout passieren. Als Abhilfe stehen Ihnen viele verschiedene Maßnahmen zur Verfügung.

Die Lösung mit dem wenigsten Aufwand ist die Komprimierung Ihrer XML-Datei im Format Gzip. Die empfangene Datei wird dann auf Serverseite automatisch wieder entpackt, bevor WordPress sie bearbeitet. So können Sie eine Kompressionsrate von 70–80 Prozent erreichen. Wenn Ihre Hosting-Einstellungen also beispielsweise 8 MByte an Upload-Größe verträgt, dann können Sie mit Kompression XML-Daten von 30–40 MByte am Stück hochschieben – und sparen dabei noch deutlich an Upload-Zeit.

Als Alternative können Sie dem WordPress auch kleine Segmente des XML statt der ganzen Datei anliefern, denn der Importmechanismus fügt die empfangenen Informationen dem aktuellen Bestand hinzu. Allerdings ist die Segmentierung nicht beliebig: Sie müssen in den Daten den Grundrahmen des WXR-Formats einhalten. Im Prinzip verteilen Sie nur die Blöcke *<item>..</item>* auf mehrere Dateien und lassen das umgebende XML gleich.

Wollen Sie diese Segmentierung nicht in Ihr Ausgabeprogramm einbauen, nutzen Sie ein Tool, etwa das in den Links erwähnte Programm von Ranger Pretzel, das sich um die korrekte Aufteilung kümmert.

Links zum Thema

- Generelle Information, wie man bestehende CMS auf WordPress konvertiert
  http://codex.wordpress.org/Importing_Content
- Erläuterungen zum WXR-Format
  http://devtidbits.com/2011/03/16/the-wordpress-extended-rss-wxr-exportimport-xml-document-format-decoded-and-explained
- Freeware-Tool, um WXR-Dateien in kleinere Einheiten aufzusplitten und so eventuell vorhandene Upload-Beschränkungen zu umgehen
  http://rangerpretzel.com/content/view/20/1
- XML-Validator von W3Schools zur Überprüfung Ihrer erzeugten Dateien
  www.w3schools.com/xml/xml_validator.asp
- Offizielle Website von WordPress
  www.wordpress.com
- Mehrsprachige Seite mit Dokumentationen zu WordPress
  https://codex.wordpress.org
- Plug-in Suicide zum Entfernen aller Beiträge, Seiten, Kommentare, Tags und Kategorien aus einer WordPress-Installation
  https://de.wordpress.org/plugins/suicide
- Freies Kompressionsprogramm Gzip
  www.gzip.org
- PHP Template Engine Twig
  http://twig.sensiolabs.org

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.

GRAFIK FÜR ENTWICKLER

Silver Surfer

IT für die Generation 65+ – eine Herausforderung für App- und Webentwickler.

Das derzeitige Altersverhältnis wird sich weiter verschieben; 2060, so das Statistische Bundesamt, wird bereits rund ein Drittel aller Deutschen der Generation 65+ angehören.

Doch ist diese Generation der heutigen Rentner nicht zu vergleichen mit der unserer Großeltern. Selbst die knapp 70-Jährigen unter uns gehören nicht zum alten Eisen, sie sind in der Regel fit, unternehmungslustig und wissensbegierig. So waren beispielsweise vor zwei Jahren 42 Prozent der Gaststudierenden an deutschen Hochschulen mindestens 65 Jahre alt.

Auch an den Volkshochschulen ist der Anteil an Kursteilnehmern über 65 Jahre deutlich gestiegen – ein beliebter Themenbereich für diese Generation: Computer und Co. So bietet beispielsweise die VHS Hamburg spezielle Computer-Seniorenkurse (Bild 1).

Doch was treibt der statistische Durchschnitts-Silver-Surfer im Internet? 2014 nutzten laut Statista.de sogar schon etwa 90 Prozent der Senioren, die über 65 Jahre alt waren, das Internet regelmäßig, um per E-Mail mit Freunden und der Familie im Kontakt zu bleiben.

Die Offenheit gegenüber neuen digitalen Techniken und dazu die relativ hohe Kaufkraft der derzeitigen Rentner machten bereits Ende des vorigen Jahrtausends die damals neu definierte Gruppe der Silver Surfer zur ertragreichen Zielgruppe auch für Internetgeschäfte.

Die Webseite Urlaub-seniorengerecht.de spricht gezielt Silver Surfer an (Bild 2)

In einer Untersuchung im Jahr 2014 haben zwei Drittel aller Personen über 65 Jahre angegeben, Bestellungen online getätigt zu haben. Hier zeigten sich jedoch die Männer mit 71 Prozent deutlich experimentierfreudiger als die weiblichen Vertreter dieser Gruppe mit einem Anteil von lediglich 60 Prozent (Tabelle 1).

Best Ager oder Silver Surfer?

Eine einheitliche Definition bezüglich des Alters von Silver Surfern gibt es nicht. So zählt Statista, das Online-Portal für Statistik, erst Personen ab 65 Jahren zu dieser Anwendergruppe. Bei Wikipedia hingegen gehören ebenfalls die sogenannten Best Ager zu den Silver Surfern, also alle Personen ab 50 Jahre.

Gerade bei der Suche im Netz peilen Silver Surfer und Best Ager zum Teil unterschiedliche Themenbereiche an. Hier passen sich die Interessen den Lebensumständen an: 50-Jährige fühlen sich in der Regel deutlich jünger jünger und stehen meist noch mitten im Berufsleben – wodurch viele von ihnen rein beruflich mit dem Internet zu tun haben.

Einmal in Rente gegangen, sind Silver Surfer ab 65 zwar nicht mehr beruflich mit den neuen Medien konfrontiert, dafür haben sie jedoch mehr Zeit, sich mit den neuen Medien auseinanderzusetzen. Mit an oberster Stelle steht die Suche nach Informationen zur Gesundheit. Auch Reisen und des Erlernen von Sprachen sind bei rüstigen Internet-Rentnern ein willkommenes Thema. Viele von ihnen haben im Ruhestand die Zeit für den langersehnten Traum-Trip und dazu (noch) das nötige Kleingeld dafür (Bild 2).

Volkshochschulen bieten seit Längerem spezielle Computerkurse für Senioren (Bild 1)

Die Site Senioren-begleiter-stuttgart.com liefert bereits auf der Startseite die wichtigsten Informationen (Bild 3)

Eva-Seniorendienste.de: Eine schlüssige Navigation hilft bei der Orientierung (Bild 4)

Muenchner-seniorenboerse.de arbeitet mit einfachen, dezent gefärbten Hintergrundflächen (Bild 5)

Doch wie macht man die neuen Medien der älteren Bevölkerungsgruppe zugänglich? Anders als für jeden 20-Jährigen, der mit Internet, Smartphone und Tablet-Computer aufgewachsen ist, sind Doppelklick und Wischgesten für ältere Medien-Nutzer keine Selbstverständlichkeit.

Erst eine transparente Nutzerführung und eine schlüssige grafische Bedienoberfläche macht es auch dem Ungeübten möglich, auf einer Webseite das zu finden, was er sucht.

Um Informationen im Netz auch für ältere Nutzer bestens zugänglich zu machen, lohnt es sich, diese möglichst übersichtlich anzuordnen. So sollte bereits die Startseite eine Übersicht über das Anliegen und die Angebote der Internetpräsenz liefern. Dabei lenken viele kleine Informationskästen, Werbeflächen oder auch eventuell sinnvoll geplante Call-to-Action-Felder ab. Kann jedoch darauf nicht verzichtet werden, sollte der Informationsblock zur Seite möglichst präsent zusammengehalten werden. Dieser kann durchaus bebildert sein, sollte aber einen ausreichenden und dabei nicht allzu langen Informationstext liefern (Bild 3).

Zudem ist es gerade für ältere Anwender hilfreich, wenn die Seitenstruktur auf den ersten Blick zu erkennen ist. Dazu müssen sich die Links zu den Unterseiten vom Rest der Seite deutlich abheben und eindeutig als solche gekennzeichnet sein, etwa durch einen farbig dezent unterlegten Kasten (Bild 4). Für einen guten Überblick punkten ruhige Hintergrundflächen, vor denen Texte gut lesbar sind. Hier eignen sich einfarbige helle Flächen am besten. Text vor einem sehr dunklen Hintergrund ist schlecht lesbar – abgesehen davon, dass schwarze Flächen nicht gerade positive Gefühle vermitteln (Bild 5).

Auffällige Muster, Grafiken oder Fotos im Hintergrund funktionieren nur bei wenig Text; hier darf nicht mehr als eine knappe Überschrift stehen. Beispiel Wir-Senioren.org: Auf dieser Internetseite ist der gutgemeinte Versuch, der Zielgruppe notwendige Informationen zu liefern, leider missglückt. Neben einem völlig chaotischen Seitenaufbau aus Kästen in verschiedenen Größen mit unterschiedlichen Inhalten tragen die Hintergründe nicht gerade zur optimalen Lesbarkeit und Orientierung bei (Bild 6).

Fließtexte funktionieren am besten, wenn sie möglichst knapp gehalten werden, da das Lesen längerer Texte am Bildschirm ermüdet. Generell ist eine serifenlose Schrift im ▶

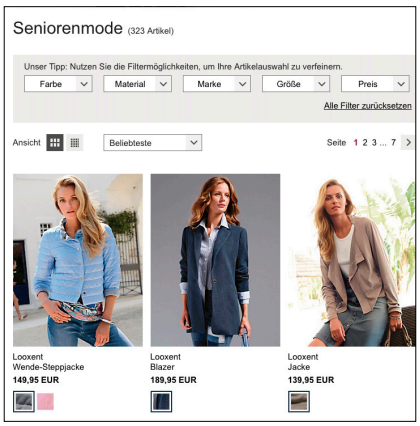
Tabelle 1: Internetaktivitäten von Personen ab 65 Jahre

	Insgesamt	Männlich	Weiblich
E-Mails schreiben/ empfangen	89,7 %	92,3 %	86,4 %
Suche nach Informationen über Waren und Dienstleistungen	84,2 %	86,1 %	81,8 %
Buchen von Reisedienstleistungen	65,2 %	70,5 %	58,6 %
Lesen von Online-Nachrichten und -Zeitungen	64,5 %	69,1 %	58,5 %
Internetbanking	42,9 %	50 %	33,9 %

Destatista.de

Internet als besser lesbar als eine Schrift mit Serifen. Dies gilt besonders bei herkömmlichen Monitoren mit einer geringeren Auflösung: Serifen, feine Linien an den Enden eines Buchstabens, können so nicht immer ausreichend dargestellt werden.

Ist die Schrift zu klein, ermüdet es den Betrachter und er wird schnell von der Seite abspringen. Wer dennoch nicht auf eine kompakte Ansicht größerer Fließtexte verzichten möchte, bietet seinen Lesern die Möglichkeit, unter verschiedenen Schriftgrößen zu wählen (Bild 7).



Mode für Senioren: Bei Peter Hahn sorgen junge Modelle für die positive Erinnerung an früher (Bild 8)

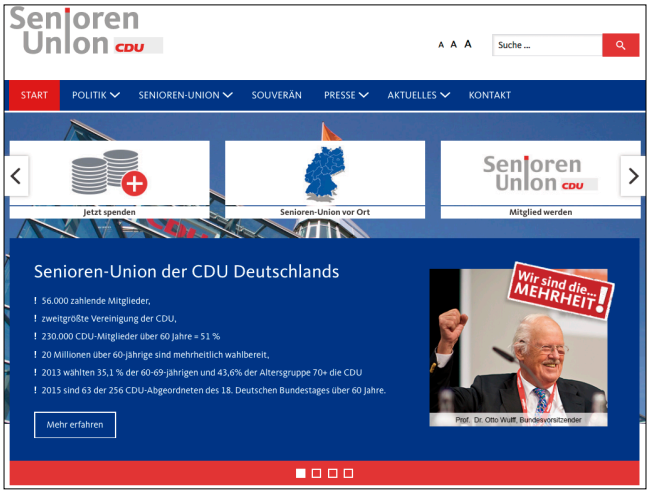
Funktion von Bildern

Bilder funktionieren dann am besten, wenn sich der Betrachter auf irgendeine Art und Weise mit dem Gezeigten identifizieren kann. Leider ist das nicht ganz so einfach: Sucht beispielsweise die silversurfende reife Dame nach einem neuen Outfit, dann sprechen nicht unbedingt alle Modeseiten dieselbe Frau an – und das ist den Webbetreibern durchaus bewusst. Zwei Beispiele hierzu: Auf Bild 8 präsentiert das Modehaus Peter Hahn seine eher konventionellen Modelle für Senioren an wesentlich jüngeren Frauen. Die Käuferin wird auf diese Weise an ihr eigenes früheres Aussehen erinnert. Das schwedische Modelabel von Gudrun Sjöden hingegen zeigt durchaus auch ältere Damen in den farbstarken Kleidern. Hier wird die selbstbewusste junggebliebene Silver-Surferin angesprochen, die sich auch mit allen altersbedingten Veränderungen ihres Aussehens attraktiv findet und eher zu ausgefallenerer Kleidung greifen möchte (Bild 9).

Soll der Betrachter hingegen etwas ganz Neues lernen, ist ein blutjunges Model auf dem Titelbild nicht gerade angebracht. Hier klappt die Identifikation nur über Bildmaterial, das auch die Angehörigen einer ähnlichen Altersgruppe zeigt. Auf der Webseite Silversurfer-rlp.de ist beispielsweise im Vordergrund sogar noch das Hörgerät des Kursteilnehmers zu erkennen. Solche Bilder schaffen beim betagten Betrachter Vertrauen in die eigene Lernfähigkeit und setzen damit mögliche Hemmschwellen bei den potenziellen Teilnehmern herab (Bild 10).

Gerade modern gehaltene Internetseiten gehen äußerst sparsam mit Animationen und Bewegtbildern um. Und das nicht ohne Grund: Je schlichter das Erscheinungsbild einer Webseite, desto mehr liegt der Fokus auf den wesentlichen Inhalten und verstärkt deren Wirkung. Ähnlich ist es mit Musik und anderen Geräuschen. Ablenkende Hintergrundmusik sollte grundsätzlich vermieden werden oder zumindest per Knopfdruck abzuschalten sein. Gibt es gesprochene Informationen, muss auf eine klare und deutliche Aussprache geachtet und Dialekt vermieden werden.

Wir-Senioren.org zeigt sehr unruhige Hintergründe, die die Bedienung nicht gerade erleichtern (Bild 6)



Auf der Site Senioren-Union.de kann der Benutzer eine individuelle Schriftgröße wählen (Bild 7)

Einschränkungen im Alter

Zwar ist es unumstritten, dass die Zahl der älteren Web- und Mobilanwender stätig steigt.

Dennoch können – müssen aber nicht – gerade bei der Altersgruppe ab etwa 65 Jahren einige Einschränkungen auftreten, die die Bedienung mobiler Geräte erschweren. Diese können im Wesentlichen in drei Gruppen gegliedert werden:

- 1. Sensorik: Die Beeinträchtigungen der Sinne umfassen das Sehen, Fühlen und Tasten sowie das Hören.
- 2. Motorik: Zudem ist die Beweglichkeit des Körpers beeinträchtigt, was sich auch auf die Fingerfertigkeit auswirkt. Durch entsprechendes Training kann dieser Prozess jedoch relativ gut eine gewisse Zeit lang herausgezögert werden.
- 3. Kognitive Veränderungen: Die dritte Gruppe der Beeinträchtigung bezieht sich auf die geistige Verfassung; so nehmen mit fortschreitenden Alter zunehmend Merk- und Koordinationsfähigkeit sowie das Reaktionsvermögen ab.

Bei Gudrunsjoeden.de stehen ältere Damen zu ihren Falten (Bild 9)

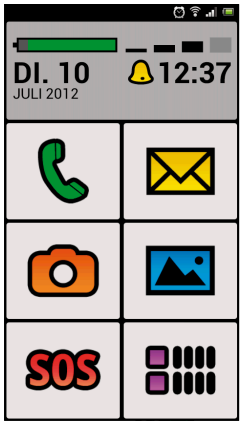
Silversurfer-rlp.de: Die Identifikation mit dem abgebildeten älteren Kursteilnehmer wird durch das gezeigte Hörgerät verstärkt (Bild 10)

Etwa 92 Prozent aller 55- bis 69-Jährigen zählten im vergangenen Jahr zu den Mobilfunkteilnehmern; gute zehn Jahre zuvor waren es mit 57 Prozent noch deutlich weniger. Gerade die intuitive Bedienung von iPhone, iPad und Co bringt die digitalen Welten auch in das Wohnzimmer – oder die Handtasche – von Senioren.

So dient das Tablet seinem betagten Besitzer als wahrer Allrounder: Telefonieren, E-Mails schreiben, im Internet surfen, Magazine oder E-Books lesen oder die letzten Urlaubsfotos betrachten; alles ist möglich.

Hinzu kommen verschiedene Bedienungshilfen, die es mittlerweile für die meisten Ausgabegeräte gibt. So kann die Schriftgröße eingestellt werden, der Kontrast angepasst und eine Sprachfunktion aktiviert werden. Zudem drängen mittlerweile spezielle Smartphones für Senioren auf den Markt, etwa das Doro 8031 (Bild 11).

Das Smartphone Doro 8031 ist einfach zu bedienen und speziell auf Senioren ausgerichtet (Bild 11)



BIG Launcher: Nur wenige Bedienelemente und knackige Farben erleichtern die Bedienung eines Android-Smartphones (Bild 12)

Trotzdem sollte auf eine schlüssige Benutzbarkeit geachtet werden, wenn die App an Senioren adressiert ist. Eine einfache Bedienung durch wenige, leicht erkennbare Steuerelemente kommt allen Nutzern entgegen, die bereits kognitive Beeinträchtigungen haben. Dabei sollten die einzelnen Elemente jeweils nur eine Funktion aufweisen und sich von anderen Bedienelementen deutlich unterscheiden.

Schaltflächen dürfen zudem nicht zu klein sein und müssen einen ausreichenden Abstand zueinander haben. Auch Scrollbalken können in der Bedienung missverständlich sein, da sie für Unerfahrene kein intuitives Signal darstellen. Hier ist es besser, die Inhalte auf mehrere Screens zu verteilen.

Um die Bedienung auch den Anwendern zu erleichtern, die sensorische Einschränkungen haben, sollte bei der Planung der App auf gute Kontraste geachtet werden. Gerade bei der Bedienung der App auf einem Smartphone sollten Bereiche auf dem Display berücksichtigt werden, die beim Halten des Geräts nicht so gut zu erreichen sind. Wie bei der Planung von Internetseiten sollte auch hier auf eine gut lesbare Schrift ohne Serifen geachtet werden. Ist ein Android-Gerät in Benutzung, hilft die App Big Launcher. Nach deren Installation erscheint anstelle der Standardoberfläche eine einfachere Bedienoberfläche, die alle Anforderungen an ein seniorengerechtes Smartphone umsetzt (Bild 12).

Fazit

Internet, Smartphone und Tablet ziehen längst nicht mehr nur jüngere Anwender in ihren Bann. Verschiedene Studien haben bewiesen, dass Silver Surfer zunehmend die digitalen Möglichkeiten ausschöpfen – sei es zur Informationssuche, Kontaktpflege oder auch zum Spielen. Dabei übernimmt letztere Möglichkeit eine nicht ganz unwichtige Rolle: Gerade für die ältere Generation ist das Spielen nicht nur eine willkommene Kurzweil, sondern trainiert nebenbei wichtige kognitive Funktionen und hält somit geistig fit. ■

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 renommierten Fachzeitschriften und verwirklichte mehrere Buchprojekte.

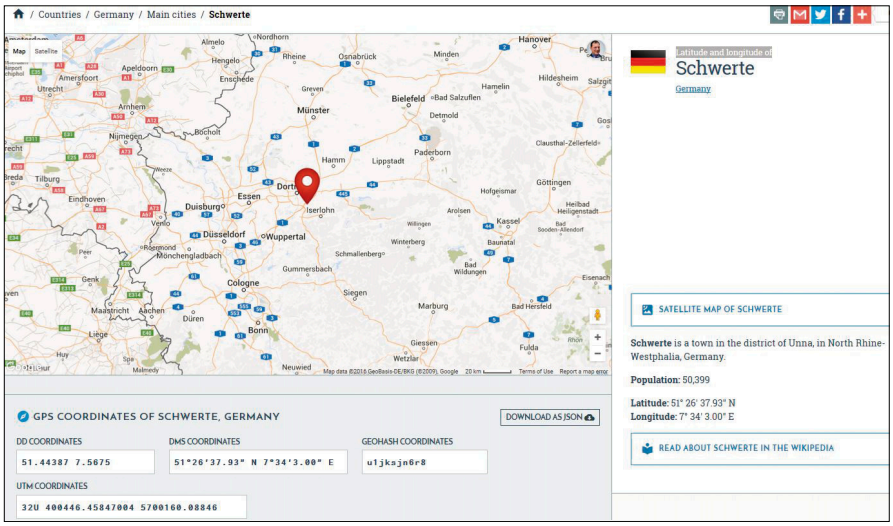
UMGANG MIT ORTSBEZOGENEN INFORMATIONEN

Geo-Informationen

Ortsbezogene Informationen werden nicht erst seit Google Maps immer wichtiger.

Vor zehn Jahren waren Geodaten, Algorithmen und Karten nur etwas für Spezialisten der Geoinformatik. Spätestens durch Google Maps und das Aufkommen von Smartphones mit eingebautem GPS wurde die Verwendung von Geodaten immer einfacher und wichtiger. Inzwischen gibt es eine Vielzahl von kostenlosen Werkzeugen, Produkten, Daten und Karten, die diese Daten für alle einfach im Browser oder Smartphone verfügbar machen. Trotzdem gibt es bei der Speicherung und Verarbeitung von Geo-Informationen einiges zu beachten.

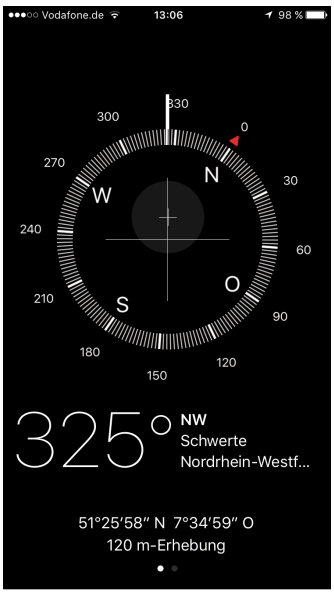
Schon bei der Darstellung des Ortes auf einer Karte oder einer Entfernungsberechnung geht das Chaos aufgrund von unterschiedlichen Formaten und Verfahren los.



Längen- und Breitengrade der Stadt Schwerte (Bild 2)

Um herauszufinden, wo man sich genau befindet, gibt es verschiedene Möglichkeiten. Oft hilft es, einfach nach dem nächsten Straßennamen zu suchen. Aber gerade in fremden Orten möchte man genauere Informationen haben. Wer ein iPhone hat, kann dort einfach die Kompassanwendung öffnen und sich seine GPS-Koordinaten anzeigen lassen (Bild 1).

Alternativ kann man auch in Google Maps seine Adresse eingeben und erhält dann im URL die Längen- und Breitenangaben als Dezimalgrad (DG) mit Punkt als Trenner. Es gibt mit UTM und GEOHASH noch weitere Darstellungsformate. Diese können Sie sich mit http://latitude.to für verschiedene Städte anzeigen lassen. Dieses einfache Beispiel zeigt, dass es nicht nur ein, sondern mehrere Formate und Verfahren gibt, um einen Ort zu bestimmen. Genauso unübersichtlich wie bei den Darstellungsformaten wird es, wenn man noch zweidimensionale Informationen wie Karten hinzunimmt (Bild 2).



GPS-Koordinaten in einem iPhone anzeigen (Bild 1)

Das richtige System

Für lange Zeit waren Systeme von Esri bei der Verarbeitung von Geodaten gesetzt, auch wenn es für bestehende Datenbanken wie Oracle, Microsoft oder DB2 Spatial-Erweiterungen gab. Eine besonders populäre Open-Source-Alternative ist die spezialisierte PostgreSQL-Version PostGIS, die seit 2001 und aktuell in der Version 2.2 für eine Vielzahl von Betriebssystemen und auch in der Cloud zur Verfügung steht. PostGIS wird von einer Vielzahl von Produkten unterstützt. Inzwischen gibt es mit MySQL, SQLite und H2 weitere Alternativen.

Ähnlich wie bei relationalen Systemen gibt es zwar mit ISO 19125 (Simple Feature Access) einen Standard für eine Architektur, der sowohl vom Open Geospatial Consortium (OGC) und der International Organization for Standardization (ISO) verantwortet wird, jedoch oft von den Produkten erweitert wurde.

Nicht nur bei der Abfrage, auch bei der Speicherung gibt es bei Geodaten Unterschiede. Hat sich im Bereich relationaler Systeme der B*-Baum für die Indizierung durchgesetzt, so ist bei Geodatenbanken der R-Baum weit verbreitet.

PostGIS verwendet als Index die spezialisierte R-tree-over-GiST-Variante (Generalized Search Tree). Für die Verarbeitung von zweidimensionalen Objekten gibt es jedoch auch Q-Bäume und für 3D-Objekte O-Bäume. Q-Bäume teilen zweidimensionale Räume rekursiv in vier Quadranten auf, weshalb diese im Englischen als Quadtrees bezeichnet werden. Eine mit dem Präfix abgewandelte Variante, der QuadPrefixTree, verwendet die Volltextbibliothek Apache Lucene.

Dessen Spatial-Modul beruht auf der Spatial4j-Bibliothek (https://github.com/spatial4j) und der JTS Topology Suite. Bei dem auf Vektoren basierten Suchalgorithmus von Lucene kann über Punkte, Rechtecke oder Kreise sowohl ein GeoSpatial-Kontext als auch ein 2D-euklidischer/kartesischer Kontext berechnet werden, um je nach Algorithmus und Gewichtung Elemente in einer bestimmten Distanz zu bestimmen.

Wir wollen das Ganze am Beispiel mit dem Apache Solr-Server ausprobieren, der auf Apache Lucene basiert. Als Ablaufumgebung wird für die folgenden Beispiele eine Java-8-Laufzeitumgebung benötigt. Wir möchten alle angebotenen Produkte im Umkreis von 100 km von 45.17614,-93.87341: Buffalo, MN anzeigen lassen. Wir starten dazu den Apache Solr-Server mit *D:\solr-5.5.0\binsolr -e techproducts*.

Danach rufen wir die Suchoberfläche mit *http://localhost:8983/solr/techproducts/browse* auf. Das zum Indizieren verwendete Schema mit den Feldern befindet sich in der Datei *C:\solr-5.5.0\server\solr\configsets\sample_techproducts_configs\conf\managed-schema*.

Alternative zu Solr

Eine Alternative zu Solr ist sicherlich ElasticSearch, das auch auf Apache Lucene basiert und Spatial-Daten unterstützt (Bild 3). Eine andere Möglichkeit, sich mit Geodatenbanken zu befassen, bietet die H2GIS-Datenbank (www.h2gis.org). Diese basiert auf der H2-Datenbank (Hypersonic 2), die wiederum eine Neuentwicklung der weitverbreiteten HSQLDB ist (Bild 4). Der Datenbankserver wird mit *java -jar h2-dist-1.2.3.jar* gestartet. Nachdem wir uns mit Benutzernamen und demselben Passwort unter *http://localhost:8082* angemeldet haben, können wir die ausgewählte Datenbank für die Spatial-Unterstützung aktivieren:

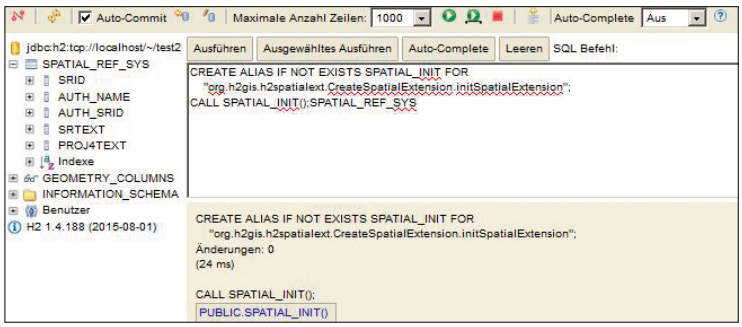
```
CREATE ALIAS IF NOT EXISTS SPATIAL_INIT FOR
"org.h2gis.h2spatialext.CreateSpatialExtension.
initSpatialExtension";
CALL SPATIAL_INIT();
```

Diesmal möchten wir die Anzahl der Straßen ausgeben, die sich in drei polygonen Bereichen befinden. Dazu legen wir für *area* und *road* die Tabellen mit einem *SPATIAL INDEX* an und befüllen diese mit Daten:

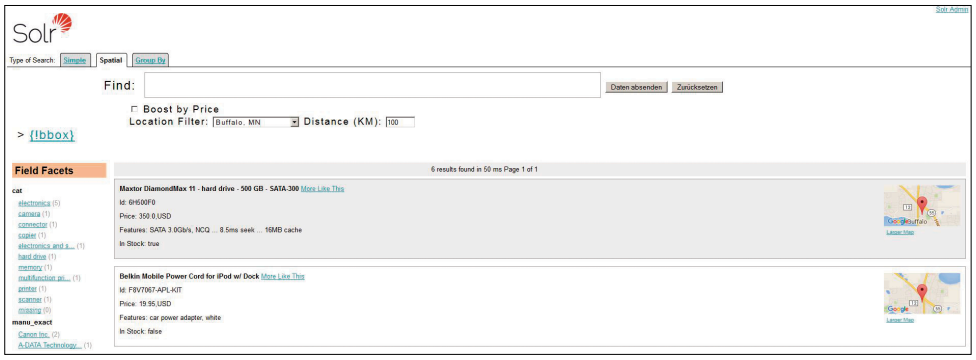
```
CREATE TABLE area(idarea INT PRIMARY KEY,
the_geom GEOMETRY);
CREATE SPATIAL INDEX myspatialindex ON area(the_geom);
CREATE TABLE roads(idroad INT PRIMARY KEY,
the_geom GEOMETRY);
CREATE SPATIAL INDEX ON roads(the_geom);

INSERT INTO area VALUES (1,
  'POLYGON((0 0, 20 0, 20 10, 0 10, 0 0))');
INSERT INTO area VALUES (2,
  'POLYGON((25 5, 40 5, 40 15, 25 15, 25 5))');
INSERT INTO area VALUES (3,
  'POLYGON((45 10, 50 10, 50 13, 45 13, 45 10))');
INSERT INTO roads VALUES (1, 'LINESTRING(2 2, 7 7)');
INSERT INTO roads VALUES (2, 'LINESTRING(15 -1, 30
13)');
```

Danach fragen wir die jeweiligen Straßen ab, die sich in den vorhandenen Bereichen befinden:

```
SELECT idarea, COUNT(idroad) roadscount
  FROM area, roads
  WHERE area.the_geom && roads.the_geom
  AND ST_Intersects(area.the_geom, roads.the_geom)
  GROUP BY idarea
  ORDER BY idarea;
```



Die Weboberfläche der H2-Datenbank (Bild 4)

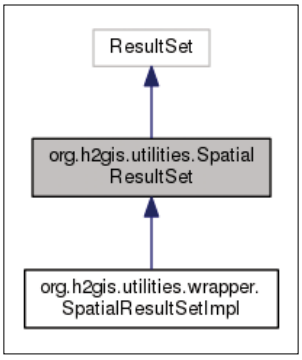


Ortsbezogene Suche mit Apache Solr (Bild 3)

Neben der Datenbank stellt H2GIS eine Geodaten-Wrapper-Schnittstelle (spatial-utilities, org.h2gis.utilities.SFSUtilities) für Java Database Connectivity (JDBC) zur Verfügung, die für H2- und PostGIS-Datenbanken verwendet werden kann. So können die meisten SQL-Befehle ohne große Anpassungen für beide Produkte verwendet werden, sodass nur der jeweilige JDBC-Treiber und die Verbin- ▶

dungsinformation angepasst werden müssen. Dazu müssen die DataSources, die Connection und das ResultSets in eine H2GIS-Klasse umgewandelt werden (Bild 5):

```
DataSource wrappedDataSource =
SFSUtilities.wrapSpatialDataSource
(originalDataSource);
Connection wrappedConnection
SFSUtilities.wrapSpatialConnection
(originalConnection);
SpatialResultSet rs = st.executeQuery
("SELECT the_geom FROM mygeomtable").
unWrap(SpatialResultSet.class);
rs.next();
Geometry myGeom = rs.getGeometry("the_geom");
```



Umwandeln von JDBC-Klassen in H2GIS-Klassen (Bild 5)

In H2HGIS ist es sogar möglich, das von der Firma ESRI für ArcView entwickelte Format für Geodaten zu verwenden. Dazu legen Sie mit *CALL FILE_TABLE('/home/user/myshapefile.shp', 'tablename');* eine Tabelle an.

Als eine alternative kleine Datenbank kommt auch SpatiaLite (https://www.gaia-gis.it/fossil/libspatialite) infrage. Diese bietet eine GIS-Erweiterung für die weitverbreitete SQLite-Datenbank an und ist wie diese in C programmiert. Es gibt hier zwar auch fertig kompilierte Versionen, jedoch sind diese für Windows und Android 4.3 auf einem etwas älteren Stand als die Linux-Versionen. Eine Spatial-Erweiterung für die Apache Derby oder H2-Datenbank ist HatBox (http://hatbox.sourceforge.net).

Wer seine Spatial-Programme in Java gern datenbankunabhängig und damit zukunftssicher schreiben möchte, für den bietet sich Hibernate Spatial an. Dieses nutzt die Java Topology Suite (JTS) und bietet Unterstützung für die Datenbanken PostgresSQL, Oracle 11g, MySQL, MS SQLServer2008 und GeoDB (H2 https://github.com/jdeolive/geodb). Da Hibernate Spatial ab der Version 5.1 Teil von Hibernate ORM ist, finden sich neue Informationen unter http://hibernate.org/orm/documentation/5.1. Tutorials und genaue Informationen darüber, welche SFS-Funktionen von welcher Datenbank unterstützt werden, finden sich bis zur Version 4.3 immer noch unter der alten Seite www.hibernatespatial.org.

Eine wichtige Werkzeugsammlung zur Bearbeitung und Darstellung geografischer Daten ist GeoTools. Diese verwendet die Java Topology Suite und implementiert die Spezifikation Simple Feature Access. Deshalb wird sie sowohl separat als auch für Web-GIS-Projekte (WFS, WMS) wie den GeoServer verwendet. Es werden eine Vielzahl von Vektorformaten (ESRI Shapefile, OGC Geography Markup Language (GML), OGC Web Feature Server (WFS), PostGIS, Oracle Spatial, ArcSDE, MySQL, SQLlite) und Raster-Formaten (OGC Web Mapping Server (WMS), ArcGrid, GeoTIFF) unterstützt. Alternativ kann man auch mit der Geospatial Data Abstraction Library (GDAL) eine Vielzahl von Geoformaten umwandeln.

Karteninformationen sind wichtig

Wenn Sie noch etwas mehr mit Karten in verschiedenen Formaten arbeiten wollen, bietet sich OrbisGIS an, mit denen Sie sowohl Karten als auch Datenbankinformationen an einer Stelle verwenden können. Dazu starten Sie den Server mit *c:\orbisgis-dist-5.1.0-SNAPSHOT-2016-03-30>orbisgis_windows.bat*. Dieser legt standardmäßig eine H2-Datenbank an.

Wenn Sie noch etwas professioneller mit Karten und Geo-Informationen arbeiten wollen, bietet sich der freie und ausgereifte GeoServer 2.8 (http://geoserver.org) an. Dieser unterstützt viele wichtige Standards, darunter Web Map Service, Web Coverage Service, Web Processing Service und Web Feature Service Interface Standard. Er kann entweder standalone oder als Webanwendung in einem Servlet-Container wie Tomcat oder Jetty installiert werden. Dann kann er mit *http://localhost:8080/geoserver* aufgerufen werden.

Fazit

Der einfachste Einstieg in das Thema GIS mit Datenbanken gelingt mit H2GIS. Das hat auch den Vorteil, dass man neben einer grafischen Oberfläche als Geo-Datenbank zur Entwicklung die schlanke H2 oder für große Installationen die ausgereifte PostGIS verwenden kann. Mit den GDAL-Werkzeugen oder GeoTools wird es einfacher möglich, geografisches Datenmaterial für die gewünschte Zielplattform zur Verfügung zu stellen. Damit sollte es in Zukunft bequemer werden, ortsbezogene Informationen zu speichern und zu verarbeiten. ■

Links zum Thema

- Spatial Extension für die H2-Datenbank
  www.h2gis.org
- OrbisGIS
  http://orbisgis.org
- Von GDAL (Geospatial Data Abstraction Library) unterstützte Formate
  www.gdal.org/ogr_formats.html
- GDAL-Werkzeuge
  http://download.osgeo.org/gdal/2.0.2
- GeoTools
  http://geotools.org

Frank Pientka
ist Senior Architekt bei der Materna GmbH in Dortmund. Er beschäftigt sich schon seit einigen Jahrzehnten mit Datenmengen, mobilen und Webanwendungen.

VDI – Vereinfachen Sie die Migration zu Windows 10 und darüber hinaus

Machen Sie sich schnell und einfach bereit für Windows-10-Endgeräte, -Anwendungen und -Desktops mit Desktop-Virtualisierung

GRATIS Whitepaper

Lesen Sie im kostenlosen Whitepaper von CITRIX®

- Optimierte Betriebssystemmigration durch Virtualisierung
- Der richtige Umstieg auf Windows 10
- Integration unterschiedlicher Endgeräte (inkl. BYOD)
- Migration der Anwendungen
- Standardisierung der Desktop-Images

Jetzt kostenlos downloaden:
http://digital.internetworld.de/citrix-wp-vdi-wump/

Ein Service von:
INTERNET WORLD Business
Ihr Whitepaper bei uns?
Anfragen an sales@nmg.de
oder Tel. 089 / 7 41 17 – 124.

BUILD 2016

Windows 10 für Entwickler

Das Kernthema der Entwicklerkonferenz war Windows 10 aus der Sicht der Entwickler.

Bereits zum fünften Mal fand Anfang April die für Microsoft wichtigste Entwicklerkonferenz Build 2016 (https://build.microsoft.com) im Moscone Center in San Francisco statt (Bild 1).

Die zentrale Botschaft lautete: Windows 10 wird, dank der kostenlosen Update-Option, vom Markt gut angenommen. Es soll nach den Vorstellungen von Microsoft ein Betriebssystem für alle Geräte sein.

Diese Botschaft ist prinzipiell nicht neu. Auch in der Vergangenheit positionierte Microsoft sein Windows immer für alle Geräte. Für den Desktop waren dies Windows 3.1, Windows 95, Windows XP, Windows NT, Windows 2000, Windows Vista, Windows 7, Windows 8 und nun Windows 10. Analog gab es mehrere Serverversionen, zuletzt Windows Server 2012 und bis hin zum Windows Server 2016. Dazu Windows CE (embedded) für weitere Consumer-Geräte. Und schließlich Versionen für Netbooks, Handys und die Xbox.

Microsoft positioniert Windows 10 als Betriebssystem für alle Geräte (Bild 2)

Windows 10 heute

Windows 10 soll wieder ein Betriebssystem für alle denkbaren Geräte werden. Dazu gehören natürlich die klassischen PCs und Notebooks, aber auch iOS- und Android-Geräte. Und auch für die mit Windows 10 neu vorgestellte HoloLens sollen sich Anwendungen erstellen lassen (Bild 2).

Microsoft verspricht sich von dieser Strategie Synergieeffekte in der Entwicklung der Systeme. Das ist sicherlich zutreffend. Dennoch hat man auch in Redmond längst erkannt, dass ein Vorgehen im Stil von »One size fits all« nicht immer optimal sein muss. Daher bietet Microsoft seinen Server schon seit Jahren auch in der Core-Version, ohne GUI, an. Diese zielt vor allem auf den Einsatz in virtuellen Umgebungen, in denen der Server seine Dienste als Arbeitspferd ganz ohne grafische Konsole verrichtet.

Die Build ist Microsofts wichtigste Entwicklerkonferenz in den USA (Bild 1)

Microsoft ließ es sich nicht nehmen, Windows 10 als Erfolgsstory hinzustellen. Bis dato kann man, laut Aussagen von Redmond, auf 270 Millionen installierte Windows-10-Geräte verweisen. Im Vergleich zur Geschwindigkeit der Verbreitung von Windows 7 wären das 145 Prozent.

Das mag stimmen, anzumerken ist aber auch, dass das Windows-10-Update kostenlos ist. Nach dem Stand vom Frühjahr 2016 hat Windows 10 bereits eine Marktanteil von 24 Prozent, Windows 7 dominiert noch mit 40 Prozent, Windows 8 hat immerhin noch 12 Prozent, die restlichen Anteile am Betriebssystemmarkt aller Geräte halten OS X, Windows XP und Windows Vista (Bild 3).

Vom Produkt zum Dienst

Als Ziel bis 2018 peilt Redmond eine Milliarde installierter Windows-10-Systeme an. Dabei soll sich Windows immer mehr vom Produkt zum Dienst wandeln, der bei Bedarf erweitert oder geändert wird. Diese Tendenz gibt es aber bereits seit Jahren und sie ist in der IT-Branche nicht unüblich.

Langjährige Windows-Anwender erinnern sich sicher noch an die früheren Versionen des Betriebssystems. Diese wurden einmalig installiert und blieben, von einzelnen Updates oder Neuerungen abgesehen, über den gesamten Nutzungszeitraum relativ unverändert.

Mit dem Patch-Dienstag führte Microsoft bereits vor Jahren die laufende Anpassung und Fehlerbehebung von Windows ein. Seither ist die Erneuerung von Windows im Monatsrhythmus gang und gäbe. Diese Tendenz wird sich wohl noch ausweiten. Microsoft verwies in dem Zusammenhang aber auch auf einen sich im Markt hartnäckig haltenden Irrtum: Das Update auf Windows 10 ist bis zum Sommer 2016 kostenlos, und dasselbe gilt für dessen nachfolgende Nutzung auf all jenen Geräten, die diese Update bis dahin voll-

zogen haben. Laut Microsoft besteht aber die Meinung im Markt, dass Windows 10 ab den Sommer kostenpflichtig sei.

Dies ist falsch. Wer bis dahin auf Windows 10 umgestiegen ist, kann es für immer kostenlos nutzen. Ob das dann aber auch für die neu erscheinenden Dienste gelten wird, ist bis dato nicht klar.

Im Sommer 2016, ein Jahr nach der Einführung von Windows 10, soll laut Aussage von Microsoft ein Update erscheinen. Die geplanten weiteren Neuerungen sind dem Link https://www.microsoft.com/en-us/WindowsForBusiness/windows-roadmap zu entnehmen.

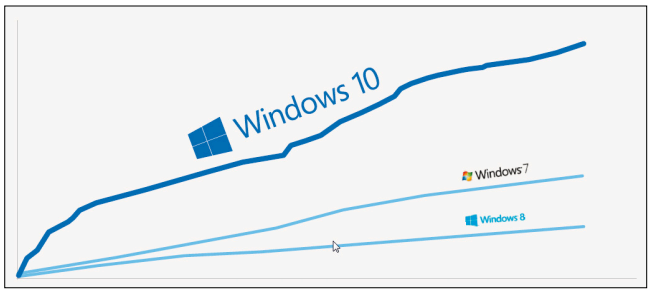
Windows 10 und der Markt

Wenngleich Windows 10 eine ansehnliche Verbreitung erfährt, zum Umsatz trägt es nicht bei. Microsoft musste daher im vergangenen Quartal einen Gewinneinbruch hinnehmen. Der Überschuss fiel gegenüber dem Vorjahrsquartal um rund ein Viertel auf 3,76 Milliarden Dollar (3,33 Milliarden Euro).

Der Erlös sank in dem Ende März abgeschlossenen dritten Geschäftsquartal um 5,5 Prozent auf 20,53 Milliarden Dollar. Die andauernde PC-Flaute hat ihre Spuren hinterlassen.

Die reinen Windows-Erlöse sanken um zwei Prozent. Gemessen am allgemeinen Rückgang der PC-Verkäufe von circa 10 Prozent ist das zwar noch hinnehmbar. Weniger erfreulich sieht es aber beim Gesamtgeschäft in dem Bereich mit Windows sowie Geräten, Spielen und der Websuche aus. Zwar ist dieser Bereich für Microsoft noch immer der stärkste Umsatzbringer. Der Umsatz beim Geschäft mit Windows sowie mit Geräten und Spielen legte jedoch nur um rund ein Prozent auf 9,46 Milliarden Dollar zu.

Auch das Auskunftssystem Cortana, ein Sprachassistent mit weiblicher Stimme, ist nach den Worten von Microsoft gut beschäftigt. Bis dato hat die Dame über 5 Milliarden Anfragen erhalten. Täglich kommt eine Million neu dazu. Dass dabei auch so manche zweifelhafte und anzügliche Anfrage von Männerseite darunter war, hat Microsoft vor einigen Monaten dazu veranlasst, Cortana etwas mehr Abwehrhilfen mitzugeben und sich gegenüber den virtuellen Belästigungen zu erwehren.



Das kostenlose Windows 10 kann, im Vergleich zu Windows 7 im gleichen Zeitraum, auf eine gute Akzeptanz bauen (Bild 3)

Die auf der Build präsentierten Neuerungen fasst Microsoft in die jeweiligen Entwicklungsschienen zusammen. Für Webentwickler sind dies native Anwendungen (Apps) in HTML und Java Script. Ferner Hosted Web-Apps, und schließlich das Entwicklungssystem Bash, das sich vor allem im Linux-Umfeld (Ubuntu) einen Namen gemacht hat.

Für Win 32 & .NET Developer bietet Microsoft die Entwicklungsplattform von Universal Windows sowie den Desktop App Converter. Diese wurde bis dato unter dem Codenamen Project Centennial geführt. (https://www.microsoft.com/en-us/download/details.aspx?id=51691):

Für die .NET-Entwicklung gibt es ferner Xamarin, Openness und Open-Source-Unterstützung sowie ein verbessertes Code-Sharing über Plattformgrenzen hinweg. Und natürlich durften auch diverse Neuerungen nicht fehlen. Dies waren Windows Hello, Windows Ink, UWP Apps, die Microsoft HoloLens Development Edition und Neues von Cortana.

Windows Hello

Mit Hello bringt Microsoft ein neues Anmeldesystem für Windows (Bild 4)

Windows Hello ermöglicht ein einfacheres Login bei Windows-10-Systemen. Anstelle der bis dato meist verwendeten Kombination aus Account-Login und Passwort treten biometrische Funktionen. Hierbei kann es sich um einen Fingerabdruck oder die Gesichtserkennung oder Iris-Scan handeln. Um die Funktionen nutzen zu können, ist allerdings eine spezielle Kamera notwendig (Bild 4).

Allerdings unterstützt nicht jedes Gerät die neue Login-Technik Windows Hello, denn sie benötigt spezielle Hardware. Eine normale Webcam, wie sie in den meisten herkömmlichen Notebooks verbaut ist, reicht dafür nicht aus. Um Gesichter zu erkennen, setzt Microsoft auf Intels RealSense-3D-Kamera.

Die Technik erlaubt eine Tiefenerkennung durch Infrarot-Kameras. Surface Pro 4, Surface Book und manche PCs bieten bereits Unterstützung für Windows Hello, und in Kürze werden weitere Geräte mit Gesichts- und Iriserkennungsfunktion folgen. Bis dato unterstützen vor allem die neueren Notebooks von Acer, Asus, Dell, HP und Lenovo Windows Hello. Intel bietet auf seiner Webseite aber auch eine eigenständige Kamera.

RealSense-Technik

Diese kann mit jedem PC verwendet werden. Intel adressiert damit aber in erster Linie Entwickler, die Hello nutzen wollen. Die RealSense-Technik eignet sich, je nach Kameraausführung, sowohl für geringe Distanzen in Räumen als auch für größere Entfernungen. Zudem können Benutzer mit dieser Technik 3D-Scans erstellen, den Fokus von Objekten in Videos nachträglich ändern und verschiedene Filter mit nur einer Fingerberührung anwenden (Bild 5).

Um Windows Hello nutzen zu können, ist dies unter den Anmelde-Einstellungen festzulegen. Wählen Sie dazu die Schaltfläche *Start* und dann *Einstellungen, Konten, Anmeldeoptionen* aus, um Windows Hello einzurichten. ▶

Windows Hello wurde bereits mit Windows 10 vorgestellt. Neu bei der Build ist nun die Möglichkeit, Windows Hello auch in anderer Software nutzen zu können. Dazu stellt Microsoft eine entsprechende Softwareschnittstelle bereit. In Zukunft können also auch Anbieter von Drittanbieter-Software Windows Hello für die Authentifizierung benutzen.

Microsoft Edge

Microsoft Edge ist der Browser von Microsoft. Er unterstützt Biometrie und erlaubt damit ein einfacheres Arbeiten mit dem Internet. Mit Microsoft Edge lassen sich Webseiten markieren oder mit anderen teilen. Statt wie heute Links in eine Mail oder ein Dokument zu kopieren und die Mail oder das Dokument dann mit Kommentaren zu versehen, erfolgt dies alles direkt in Edge. Die Edge-Nachrichten lassen sich ferner auch an weitere Empfänger oder zu OneNote oder OneDrive versenden.

Tastatur und Maus zählen nach wie vor zu den Haupteingabemedien. Daneben hat sich die Sprachsteuerung einen Nischenanteil erkämpft.

Ein dritter Weg ist die Eingabe von Information über Stifte. Diese wurden bereits in früheren Geräten wie Tablets oder vergleichbaren Benutzermedien verwendet. Abgesehen von einzelnen Nischen konnten sie aber nie einen nennenswerten Marktanteil erobern. Mit Windows Ink versucht Microsoft, ein Comeback dieser Technik einzuleiten.

Laut Aussagen von Microsoft reichen schon zwei Zeilen Code, um Windows Ink in bestehende Anwendungen zu integrieren. Zusammen mit Windows Ink hat Microsoft auch ein digitales Lineal vorgestellt. Dieses soll die Stifteingabe vereinfachen, um beispielweise Linien damit zu generieren.

Für die zusammen mit Windows 10 vorgestellte HoloLens soll es nun auch weitere Entwicklerunterstützung geben. Die Microsoft HoloLens Development Edition. Sie ist vorerst aber nur in den USA und Kanada verfügbar. In Zukunft solle jede UWP-Anwendungen auch direkt auf der HoloLens ausgeführt werden können.

HoloLens präsentiert dem Anwender Bilder und Diagramme in hochauflösender Qualität direkt auf den Bildschirm der Brille. Hinzu kommen Klangeindrücke durch Surround Sound. Microsoft HoloLens ermöglicht seinen Nutzern damit eine neue Erfahrung bei der Erstellung und dem Zugang zu Informationen sowie in den Bereichen Unterhaltung und Kommunikation. Ergänzend dazu steht der neue Großbildfernseher Surface Hub. Die Nutzer können sich die Anzeigen auf 55 Zoll sowie 84 Zoll einblenden lassen.

Die Intel RealSense-Kamera F200 eignet sich am besten für geringe Distanzen (Bild 5)

Mit der HoloLens lassen sich Bilder direkt und unmittelbar präsentieren (Bild 6)

Die Anwendergruppe von HoloLens und Surface Hub sind die Unternehmen und der Einsatz in Konferenzzentren und bei Meetings oder Präsentationen (Bild 6).

Erneuerung von Cortana

Erneuert wird auch Cortana. In Zukunft solle es mit proaktiver Unterstützung des Anwenders, falls er es wünscht, glänzen. Ferner kann Cortana in Zukunft auch in bestehende Anwendungen integriert werden.

Der Sprachassistent soll Eingang finden in bestehende Anwendungen und Social-Web-Plattformen, wie etwa Netflix, Xing oder anderen. Cortana hilft dem Anwender bei verschiedenen Aufgaben im Umgang mit Windows. Hierzu wurde dem Sprachassistenten Lernfähigkeit mitgegeben. Das System soll die ihm gestellten Aufgaben vorausschauend behandeln und Empfehlungen aussprechen.

Damit gehen die Funktionen von Cortana über die von bestehenden Sprachassistenten hinaus. Cortana erinnerte seinen Anwender an Termine und kann beispielsweise Querverknüpfungen zwischen Informationshäppchen und Notizen selbstständig aufbauen. Ferner ist Cortana in der Lage, nach Dokumenten oder Informationen zu suchen.

Linux-Subsystem for Windows

Bash ist ein Entwicklungswerkzeug (Shell), das im Linux-Umfeld eine gute Verbreitung erfahren konnte. Nun steht es auch für Windows zur Verfügung. Hierzu nutzt Microsoft ein Linux-Subsystem for Windows, in dem die Bash ausgeführt wird. In dem Subsystem der Bash-Shell lassen sich dann native Linux-Skripts ausführen.

Die Bash ist eine kommandozeilenbasierte Umgebung, ähnlich wie es Microsoft in seiner PowerShell-Umgebung bereitstellt. Dabei fungiert Linux als Basissystem, in dem die Bash ausgeführt wird. Dies ermöglicht die Ausführung von Bash Shell-Skripts und weiteren Anwendungen, die auch unter Ubuntu-Linux ausgeführt werden. Dabei gibt es allerdings einige Einschränkungen.

Es funktioniert nicht mit Serversoftware und grafisch orientierten Programmen. Die in Windows 10 eingebettete Bash-Umgebung wendet sich vor allem an Entwickler, die ihre

Kommandozeilen-Skripts unter Windows nutzen wollen. Dabei erhalten die Utilities Zugriff auf das Windows-Dateisystem. Die Steuerung von Windows-Programmen durch Bash ist allerdings nicht möglich.

Um Bash unter Windows nutzen zu können, müssen Sie den Build 14316 von Windows 10 verwenden. Es läuft nur in Verbindung mit den 64-Bit-Versionen von Windows. Außerdem muss die Insider Preview aktiviert werden. Um das Windows Subsystem für Linux für Windows zu aktivieren, öffnen Sie das Control Panel (Settings App) und darin unter *Update* und *Sicherheit* den Entwicklermodus (Bild 7).

Für den klassischen Windows-Desktop-Entwickler ist wohl der Desktop App Converter besonders interessant. Er erlaubt die Transformation von klassischen Windows-Anwendungen (UWP) in Apps. Dies passiert durch die Kapselung (Sandboxing) der Anwendungen.

Laut Microsoft sollen damit die traditionellen Anwendungen in moderne Umgebungen überführt werden. Hierbei gilt es natürlich zu bedenken, dass, wenngleich das Sandboxing seinen Zweck erfüllen mag, Apps oftmals neu designt werden sollten, um in der neuen Umgebung ihre Vorteile ausspielen zu können. Die transformierten Apps lassen sich dann auch über den Windows App Store vertreiben.

Microsoft Cognitive Services

Neues wagt Microsoft mit seinen Cognitive Services. Sie unterstützen etwa den Umgang mit Windows und Anwendungen über bestehende Grenzen hinaus. Durch Cognitive Services sollen Gesichter und Stimmungen erkannt werden.

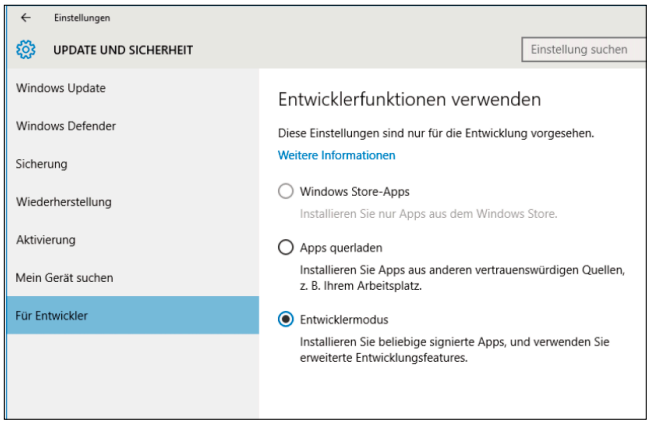
Für zukünftige Webshops oder Point-of-Sale-Systeme mag diese hilfreich sein. Wenn beispielweise der Interessent bei einem ihm präsentierten Produkt freudig aussieht, so kann der Webshop-Betreiber davon ausgehen, dass das präsentierte Produkt in die richtige Richtung geht.

Die Cognitive Services sind, wie der Name sagt, eine Sammlung an Diensten. Derzeit sind dies rund 20 unterschiedliche Services. Eingeschlossen sind zum Beispiel Bilderkennung und Analyse. Durch Entwicklungshilfen lassen sich diese Dienste auch in eigene Anwendungen integrieren.

Als Bot bezeichnet man Hintergrundprogramme, die Routinearbeiten ausführen. Meist werden dazu mehrere Rechner mit Internetanschluss benutzt. Bösartige Bots werden beispielsweise für DOS-Attacken benutzt. Bots müssen aber nicht immer bösartig sein. Webcrawler von Internet-Suchmaschinen, die selbsttätig Webseiten besuchen, wobei sie den vorhandenen Links folgen, werden ebenso als Bot bezeichnet. Technisch betrachtet handelt es sich bei Bots um die Verteilung von Aufgaben und parallele Bearbeitung durch mehrere unabhängige Instanzen (Rechner) unter einer zentralen Steuerung. Microsoft will sich die Technik nun nutzbar machen und bietet in Zukunft ein Framework für die Nutzung von Bots an.

Xamarin und Visual Studio 15

Neuerungen gibt es auch für Microsofts Entwicklungspaket Visual Studio. Es soll zu einem universellen Entwicklungskit für alle Geräte und Plattformen ausgebaut werden. Dazu gehören auch Android-Geräte und iOS-Systeme. Zu den Verbesserungen der kommenden Version gehört ein schnelleres Setup – in der Vergangenheit konnte es passieren, dass sich die Installation von Visual Studio bis zu einen Tag lang hinzog. Auf Xamarin haben die Entwickler von .NET-Anwendungen lange gewartet. Die Entwicklungshilfen von Xamarin werden im Update 2 für Visual Studio 2015 enthalten sein. Dann können die Visual-Studio-Entwickler iOS-Anwendungen im Kontext des Microsoft-Entwicklungskits erstellen.



Das Windows Subsystem für Linux erlaubt die Ausführung von Bash-Kommandos unter Windows (Bild 7)

Office-365-Integration

Integration war ein anderes Schlagwort, das auf der Build fiel. Lösungen können aus Office heraus auf Graph zugreifen, Dokumente holen, verarbeiten und wieder über Graph in OneDrive legen. Microsoft Graph ist ein API, mit dem Entwickler Zugriff auf Daten auf den Microsoft-Servern erhalten. So können Anwendungen zum Beispiel Benutzer, Gruppen, E-Mails, Kalender, Kontakte, Dateien, Aufgaben, Personen oder Notizen abrufen oder einstellen. Ein anderes Beispiel ist Zapier. Diese Lösung ist in der Lage, über Graph Excel-Dateien, die auf OneDrive liegen, zu bearbeiten.

Fazit

Es gab Stimmen, die ob der wenigen Neuheiten enttäuscht waren. Sieht man sich diese aber genauer an, zeigt sich ihre immense Tragweite. Dass mit Universal Windows Apps und den kostenlosen Tools von Xamarin jetzt Cross-Plattform-Entwicklung für alle .NET-Entwickler in greifbare Nähe rückt, ist schlicht der Hammer. ■

Johann Baumeister

hat Informatik studiert und besitzt langjährige Erfahrung in der Entwicklung, Anwendung und dem Rollout von IT-Systemen. Außerdem ist er als Autor für zahlreiche IT-Publikationen tätig.

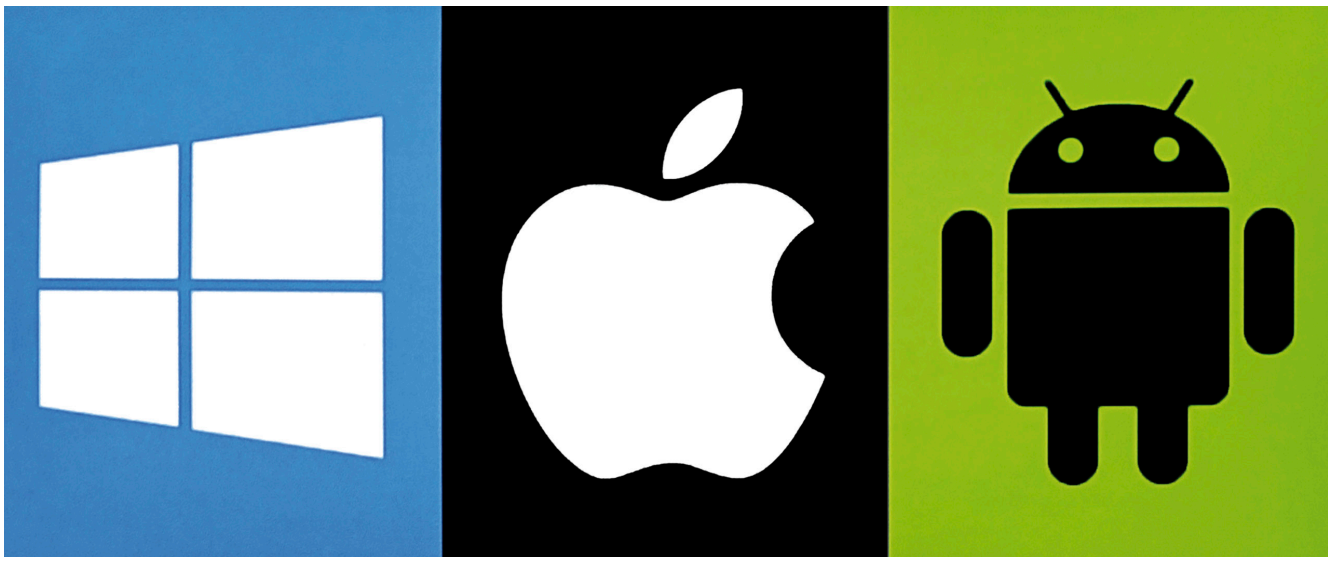


Bild: Shutterstock / Rose Carson

MEHR SICHERHEIT FÜR DIE ENDGERÄTE

Android, iOS oder Windows Mobile

Welches mobile Betriebssystem sich am besten für den Unternehmenseinsatz eignet.

Bring Your Own Device (BYOD) ist längst in vielen Unternehmen Alltag. Der Einsatz verschiedenster Smartphones und Tablets, die zunehmende private wie geschäftliche Nutzung sowie immer neue, auf die mobile IT ausgelegte Angriffsszenarien – die Nutzung der mobilen Geräte stellt die IT-Verantwortlichen vor einige Herausforderungen. Vor allem die Sicherheit gewährleisten zu müssen bringt viele Administratoren ins Schwitzen.

Laut einer Studie der Analysten von IDC wird fast jeder zweite Sicherheitsvorfall im Zusammenhang mit mobilen Technologien von Anwendern verursacht. Umso wichtiger ist es, dass man sich als Unternehmen für das richtige mobile Betriebssystem entscheidet und die Nutzung der Geräte so gut wie möglich regelt.

43 %
der Sicherheitsvorfälle mit mobilen Geräten gehen auf das Konto von Anwendern.

Quelle: IDC

Wenn man sich die Marktanteile mobiler Betriebssysteme ansieht, dann macht bei den meisten Unternehmen vor allem ein System das Rennen: Laut dem Mobility Index Report von Good Technology vom dritten Quartal 2015 läuft auf 66 Prozent der weltweiten Firmen-Smartphones Apple iOS. Android kommt auf einen Marktanteil von 31 Prozent. Noch deutlicher sind die Zahlen bei den Tablets: Hier kommt iOS auf satte 71 Prozent, gefolgt von Android mit 21 Prozent. Mit Windows laufen nur 8 Prozent der Firmen-Tablets.

Android, Blackberry, iOS oder Windows Mobile? Welches mobile System eignet sich nun aber am besten für den Unternehmenseinsatz und sorgt wirklich für die meiste Sicherheit? Und wie verwaltet man das Sammelsurium an Geräten und Betriebssystemen?

Dass viele Unternehmen bei der mobilen Nutzung vor allem auf Geräte von Apple setzen, hat auch mit der Sicherheit zu tun. Denn die ist bei Android ein großer Schwachpunkt. Aufgrund der starken Verbreitung im privaten Umfeld ist Android ein beliebtes Angriffsziel. Eine Auswertung der Antiviren-Experten von Kaspersky Lab hat ergeben, dass vergangenes Jahr 7 Prozent der von Kaspersky-Produkten in Unternehmen geblockten Exploits – Schwachstellen in Software – Android betreffen. Andere mobile Betriebssysteme tauchen in der Statistik gar nicht erst auf. Die Zunahme von Exploits, die sich gegen verwundbare Android-Apps richten, zeugt auch von einem gesteigerten Interesse der Cyberkriminellen an Unternehmensdaten.

Das Hauptproblem sind allerdings die Mitarbeiter: »Mobile Malware ist auch heute noch auf manuelle Mithilfe angewiesen, sodass Mitarbeiteraufklärung und eine starke Mobile-Device-Management-Lösung oder ein Antiviren-Schutz ein sehr hohes Maß an Sicherheit bieten«, so Christian Funk, Leiter des Forschungs- und Analyse-Teams bei Kaspersky.

Wie erwähnt kommt eine IDC-Studie zu dem Ergebnis, dass viele Sicherheitsvorfälle durch Anwender verursacht

werden. Nach Angaben der befragten deutschen IT-Entscheider gehen 43 Prozent der Sicherheitsvorfälle auf das Konto der Nutzer mobiler Geräte. Häufig ist der allzu sorglose Umgang mit den Geräten die Ursache. So verloren in den vergangene zwei Jahren 30 Prozent der Unternehmen ein Smartphone mit Firmeninformationen – 10 Prozent sogar öfter als einmal. Jeder zweite IT-Entscheider hält die Absicherung mobiler Geräte für die derzeit größte Herausforderung.

Sicherheitslücke Apps

Für IT-Verantwortliche ist vor allem die Sicherheit von Smartphone- und Tablet-Apps ein Problem. Das gilt insbesondere dann, wenn Anwender durch die IT nicht freigegebene Apps aus App-Stores wie Google Play auf ihr Gerät laden. Laut der IDC-Studie zählen für knapp ein Drittel der befragten IT-Verantwortlichen unsichere Apps zu den drei größten Sicherheitsrisiken bei mobilen Geräten.

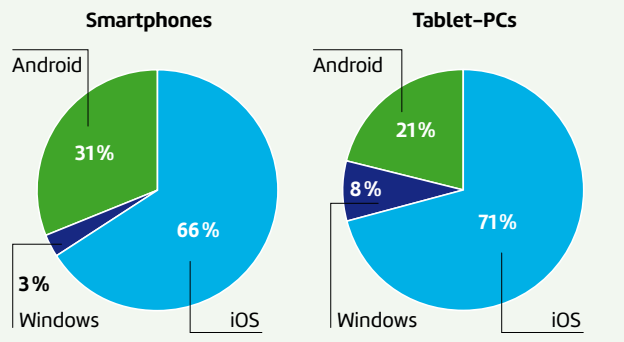
Wie sieht es mit der Sicherheit der App-Stores aus? Android-Entwickler Google sieht hier vor allem die Nutzer in der Pflicht: Wer eine App aus dem Google Play Store installiert, muss zustimmen, wenn diese auf bestimmte Bereiche des Geräts zugreifen will. Passt man hier nicht auf, dann lädt man sich ganz schnell ein Schnüffel-Tool auf sein Gerät. Unter iOS ist die Gefahr, sich eine Schadsoftware einzufangen, etwas geringer. Apple überwacht seinen App-Store restriktiver. Jede App und jedes Update wird vor der Freigabe einer gründlichen Kontrolle unterzogen.

Hier sind Blackberry und Windows Mobile eindeutig im Vorteil – in deren App-Stores herrscht nämlich gähnende Leere. »Fragen Sie doch einfach nach Business-Anwendungen für diese Plattformen, sie werden mit leeren Händen dastehen«, fasst Carsten Mickeleit, CEO des Enterprise-Mobility-Anbieters Cortado, das Angebot zusammen. Für Kriminelle rentiert es sich einfach nicht, für diese Systeme Schadsoftware zu programmieren.

In 10 % der Unternehmen gibt es mindestens ein gefährdetes Mobilgerät, das auf Unternehmensdaten zugreift.

Quelle: Mobileiron

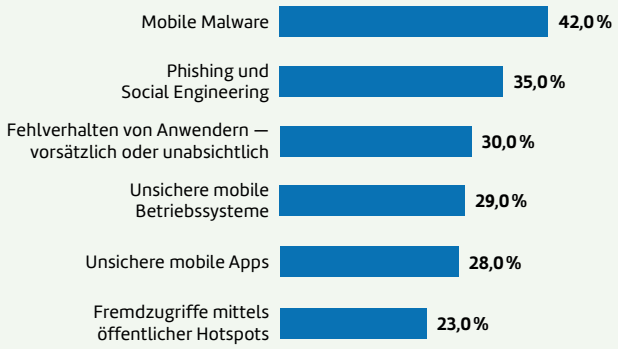
Marktanteile in Unternehmen



Apple ist führend: iOS läuft auf zwei Drittel aller weltweiten Firmen-Smartphones. Bei Tablets sind es sogar 71 Prozent.

web & mobile developer 7/2016 Goof Technology, Mobility Index Report Q3/2015

Risiken bei mobilen Geräten



Mobile Malware ist nach Ansicht von IT-Verantwortlichen die größte Gefahr beim Einsatz mobiler Geräte in Unternehmen.

web & mobile developer 7/2016 IDC-Studie »Mobile Security in Deutschland 2015«

Und das wird auf absehbare Zeit auch so bleiben: »Die Startschwierigkeiten von Windows Phone, jetzt Windows Mobile, haben nicht nur dazu geführt, dass Microsoft spät am Markt ist, sondern dazu, dass Anwendungsentwickler bis heute nicht sicher sein können, ob sie nicht in kurzer Zeit ihre Entwicklung von vorne beginnen müssen, weil Microsoft schon wieder das Betriebssystem umgebaut hat«, so Carsten Mickeleit weiter. Das trifft nicht nur auf seriöse App-Entwickler zu, sondern auch auf Programmierer von Schadsoftware.

Klaus Düll, Geschäftsführer und Experte für Mobile Device Management (MDM) beim Business-Mobility-Anbieter Pretioso, sieht in Windows Mobile dagegen durchaus Potenzial: „Microsoft hat in den letzten drei Jahren für Unternehmenskunden in der Mobility mehr richtig gemacht als Apple." Auch laut Sahin Tugcular, Geschäftsführer beim MDM-Anbieter AppTec360, hat Windows Mobile durchaus eine Zukunft im Unternehmensbereich: »Man hört in den Kundengesprächen immer mehr, dass die Firmen Windows 10 Mobile nutzen wollen, um bei den Client-Betriebssystemen eine homogene IT-Landschaft zu schaffen.«

Eine Möglichkeit, um die Sicherheit von Apps zu erhöhen, sind firmeneigene App-Stores. IT-Abteilungen richten App-Stores ein, die ausschließlich den Mitarbeitern zur Verfügung stehen. Darin liegen die Apps, die von der IT für die Nutzung freigegeben wurden. Der Zugriff auf den firmeneigenen App-Store kann sowohl über mobile Geräte erfolgen, die das Unternehmen herausgegeben hat, als auch über die eigenen Geräte der Mitarbeiter. Die Analysten von Gartner prognostizieren, dass 2017 25 Prozent aller Unternehmen einen eigenen App-Store haben werden.

Welches mobile Betriebssystem ist nun das sicherste? Hundertprozentige Sicherheit gibt es nicht, das ist klar, aber: »Ich gehe davon aus, dass ein gemanagtes Blackberry im Hinblick auf Sicherheit das absolute Maximum darstellt«, sagt Carsten Mickeleit von Cortado. »Doch Sicherheit ist immer ▶

Interview

»Die monopolistische Blackberry-Welt ist passé«

Stratos Komotoglou, Senior Manager EMEA Product & Field Marketing beim EMM-Anbieter Mobileiron, spricht mit **web & mobile developer** darüber, welche mobilen Betriebssysteme sich für Unternehmen eignen und wie es um die Sicherheit der Systeme steht.

web & mobile developer: *Android, Blackberry, iOS oder Windows Mobile. Welches System eignet sich für den Unternehmenseinsatz?*

Stratos Komotoglou: Das lässt sich nicht pauschal sagen, da der Einsatz eines Betriebssystems von vielen Faktoren abhängt. Tatsächlich wird in praktisch allen Unternehmen schon mit zahlreichen Mobilgeräten gearbeitet. Und oft ist der Unternehmenseinsatz nicht geregelt beziehungsweise befinden sich die Konzepte in der ersten Phase. Die Regelung des Mobilgeräteeinsatzes im Unternehmensumfeld sollte oberste Priorität haben. Ganz gleich welches oder wie viele Betriebssysteme ein Unternehmen verwenden will – die Grundlage sollte ein Enterprise-Mobility-Management-System bilden.

»Unternehmen tun gut daran, ihre Mitarbeiter in Sachen Mobilgerät nicht zu stark zu reglementieren.«

web & mobile developer: *Wie sieht es mit Windows Mobile und Blackberry aus? Spielen die im Business-Umfeld überhaupt eine Rolle?*

Komotoglou: Vor zehn Jahren, als mobile IT praktisch ein Synonym für mobile E-Mail war, gab es eigentlich für verantwortungsbewusste Unternehmen nur die Blackberry-Lösung. Aber diese monopolistische Blackberry-Welt ist längst passé. Blackberry ist keine Alternative, Windows Mobile dagegen ist ein ernst zu nehmender Protagonist im Markt, vor allem in der Version Windows 10, mit der das mobile Betriebssystemmodell richtungsweisend für die gesamte Betriebssystemwelt wird.
Ein Konzern wie der weltweit agierende schweizerische Rückversicherer Swiss Re ersetzt beispielsweise gerade Blackberry durch Windows Mobile , lässt aber auch iPhones und künftig vielleicht sogar Android-Geräte zu.

web & mobile developer: *Wenn Sie ein Unternehmen fragen würde, welches System es nehmen soll, was würden Sie empfehlen?*

Komotoglou: Das hängt zum einen vom Use Case und zum anderen auch vom Budget ab. Viele Unternehmen sehen Windows Mobile und Windows 10 als Schlüssel für mobile Initiativen, da man ja bereits zahlreiche Microsoft-Lösungen einsetzt. Unternehmen, die sehr viele Prozesse mit Apps mobilisiert haben, setzen beispielsweise iOS oder Android ein, da die App-Entwicklung und die Möglichkeiten hier weiter sind. Gleichwohl tun Unternehmen gut daran, ihre Mitarbeiter in Sachen Mobilgerät nicht zu stark zu reglementieren. Hier weise ich noch einmal auf das Beispiel Swiss Re hin. Das offizielle mobile Betriebssystem ist Windows Mobile, aber zugelassen ist auch iOS, und über die Einbeziehung bestimmter Android-Versionen denkt man zumindest nach.

Foto: Mobileiron

Stratos Komotoglou
Senior Manager EMEA Product & Field Marketing, Mobileiron
www.mobileiron.com/de

web & mobile developer: *Für die Nutzung mobiler Geräte und für Bring Your Own Device kommt man um ein Mobile Device Management mit Application-Management nicht herum. Gibt es bei der Eignung für MDM überhaupt relevante Unterschiede zwischen Android, iOS und Co.?*

Komotoglou: MDM-Systeme, oder besser EMM-Systeme, haben gerade die Aufgabe, etwaige Ecken und Kanten von mobilen Betriebssystemen für den Einsatz in Unternehmen zu glätten. Daher kann man sagen, dass uns als EMM-Spezialist die von Ihnen angesprochene Eignung nur insofern interessiert, als sie eine Herausforderung für uns als EMM-Hersteller darstellt. Klar, bei Android war die Herausforderung für Mobileiron in den letzten Jahren größer als bei iOS. Aber wir haben sie mit dem Android-for-Work-Modul so gelöst, dass unsere Kunden auch Android-Umgebungen im Unternehmen allein oder im Zusammenspiel mit iOS und/oder Windows Mobile sicher und zentral verwalten können. So hat beispielsweise das Schienenlogistik-Unternehmen TXLogistics sein Blackberry-Universum durch eine Samsung-Knox-Lösung mit Android abgelöst.

»Windows Mobile ist ein ernst zu nehmender Protagonist im Markt.«

web & mobile developer: *Wo stehen deutsche Unternehmen derzeit beim Thema Enterprise Mobility?*

Komotoglou: Na ja, dass kaum einer der Unternehmensanwender weiß, worum es bei EMM geht, diese Vermutung halte ich für nicht richtig. Wir als Mobileiron haben jede Menge Anwender, die zeigen, wie es richtig gemacht wird. Aber sicher, Sie haben schon Recht, es gibt noch viel zu tun. Eine große Zahl von Mobilgeräten, die Zugriff auf Unternehmensdaten haben, sind vermutlich immer noch nicht unter der Kontrolle eines EMM-Systems. Die Firmen sollten aufwachen. Und wenn ich manchmal höre, die Daten, die mobil verarbeitet werden, seien doch nicht so wichtig, dann kann ich nur staunen. Viele Unternehmen unterschätzen immer noch den Wert ihrer Daten für den Geschäftserfolg.

relativ und Unternehmen müssen die richtige Balance zwischen Sicherheit, Anwenderkomfort und vor allem Produktivität finden«, so Mickeleit weiter. Für Klaus Düll von Pretioso ist »Samsung Android die einzige Lösung, die heute feingranulare Konfigurationen erlaubt«.

Stratos Komotoglou, Senior Manager EMEA Product & Field Marketing beim Enterprise-Mobility-Anbieter Mobileiron, sieht das Ganze kritischer: »Bei der Durchsetzung von Sicherheitsrichtlinien würde ich mich auf kein mobiles Betriebssystem allein verlassen.« Laut Komotoglou kommt ein Unternehmen um den Einsatz eines Enterprise Mobility Managements (EMM) nicht herum. Nur damit sei eine umfangreiche Kontrolle von Smartphones und anderen Mobilgeräten möglich.

Enterprise Mobility Management

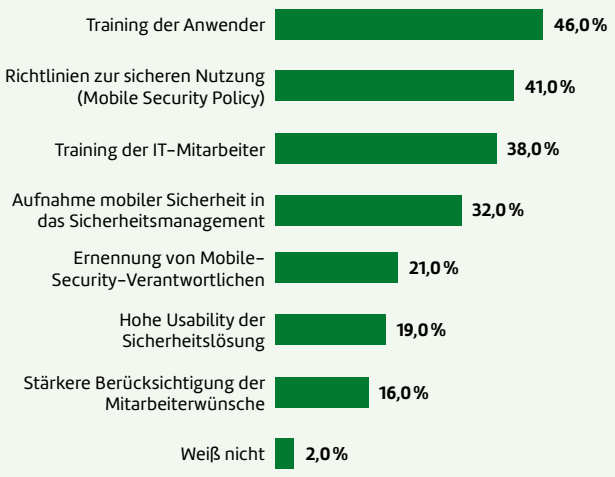
EMM bezeichnet einen umfassenden Ansatz, um den Mitarbeitern die Nutzung mobiler Geräte zu ermöglichen und diese Geräte abzusichern. Zudem soll ein EMM deren Integration in die Firmen-IT sicherstellen. EMM-Lösungen sind meist eine Kombination aus Mobile Device Management (MDM), Mobile Application Management (MAM) und Mobile Information Management (MIM):

- **Mobile Device Management:** Das MDM übernimmt die Verwaltung des mobilen Endgeräts durch die zentrale IT-Administration. Dazu gehört, das Gerät vor dem Zugriff Dritter zu schützen. So lässt sich ein Smartphone oder Tablet bei Verlust per Remote deaktivieren und alle darauf abgelegten Daten lassen sich löschen.
- **Mobile Application Management:** Das MAM regelt, auf welche Anwendungen die Nutzer zugreifen dürfen. Es gewährleistet, dass die Installation und der Zugriff auf Apps – ob Unternehmens-Apps oder Apps aus öffentlichen App-Stores – nur nach den Unternehmensrichtlinien erfolgen.
- **Mobile Information Management:** Das MIM behält die Kontrolle über die Firmendaten. Es sorgt dafür, dass nur von der Firmen-IT freigegebene Anwendungen auf Firmendaten zugreifen – und dass die Datenübertragung, etwa von der Cloud und zur Cloud, verschlüsselt erfolgt.

Häufig kommen auch Container-Lösungen zum Einsatz. Damit lassen sich die Apps und die Daten eines Unternehmens auf dem Mobilgerät in einer geschützten Umgebung verwalten. Container ermöglichen auf diese Weise eine Trennung von privatem und geschäftlichem Bereich. Nur wenige Unternehmen sehen es gern, wenn sensible Firmendaten im selben Verzeichnis liegen wie das letzte Urlaubs-Selfie des Mitarbeiters.

Doch ist jedes mobile Betriebssystem gleich gut geeignet für den Einsatz mit einer EMM-Lösung? Hier gehen die Meinungen auseinander – und je mehr Anbieter man fragt, desto mehr Meinungen hört man. »Die Unterschiede der Eignung für eine Mobility-Lösung sind sehr groß, da die Anbieter der mobilen Betriebssysteme sehr unterschiedlich leistungsfähige APIs bieten. Hier ist die Rangfolge derzeit wie folgt: Samsung Android, Windows Phone, iOS, Android, Blackberry«, so Klaus Düll von Pretioso. Für Carsten Mickeleit von Cortado ist iOS führend: »iOS 9 liegt gefolgt von Android mit Android for Work ganz klar vorn. Die Granularität der Möglichkeiten ist hier unschlagbar.« Ähnlich sieht es Sahin Tugcular von AppTec360: »Aktuell lässt sich iOS am besten verwalten.«

Verbesserung der Mobile Security



Organisatorische Maßnahmen zur Verbesserung der Sicherheit im Unternehmen.

web & mobile developer 7/2016 IDC-Studie »Mobile Security in Deutschland 2015«

In rund 53 % der Unternehmen gibt es mindestens ein Gerät, das die Sicherheitsrichtlinien der Firma verletzt.

Quelle: Mobileiron

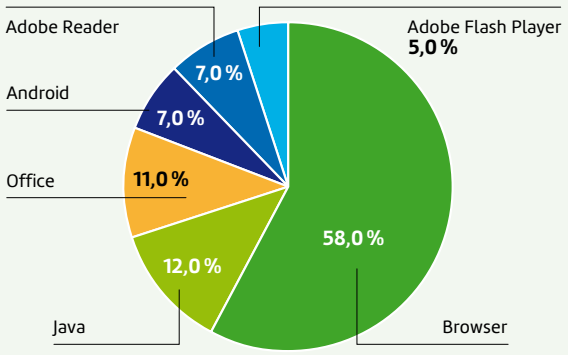
EMM-Lösungen

Obwohl mobile Geräte in Unternehmen weitverbreitet sind, spielt das Thema EMM bislang eine eher untergeordnete Rolle. »Wir sind immer noch in einer sehr frühen Phase der Enterprise Mobility, noch konzentrieren sich viele auf das Absichern von mobilen Geräten und vielfach ist die E-Mail-App die einzige Business-Anwendung«, sagt Carsten Mickeleit von Cortado. »Doch für Unternehmen ist jetzt, mit der Verfügbarkeit von iOS 9 und Android for Work, der ideale Zeitpunkt für den Einstieg, denn jetzt kann man ohne komplexe Klimmzüge ein Maximum an Sicherheit gewährleisten. Kombiniert mit der richtigen Enterprise-Mobility-Lösung, die in der Lage ist, dies zu nutzen und mit der Rechte-, Datei- und Druckinfrastruktur zu kombinieren, war Enterprise Mobility noch nie so einfach, sicher und produktiv umzusetzen wie heute«, so Mickeleit weiter.

Die Analysten von IDC empfehlen, das Thema Mobile Security nicht isoliert, sondern als Teil des gesamten IT-Konzepts zu betrachten. Unternehmen beschäftigen sich seit vielen Jahren mit der Absicherung ihrer Infrastruktur, von den Desktops-PCs bis hin zu den Servern. Sicherheitskonzepte für mobile Geräte sollte daher ein zentraler Bestandteil des unternehmensweiten IT-Sicherheitskonzepts sein.

Unternehmen, die auf eine EMM-Lösung setzen möchten, stehen vor einem Dschungel an Produkten. Der Markt ist ▶

Exploits in Attacken



Android als Angriffsziel: 7 Prozent der 2015 von Kaspersky-Produkten blockierten Exploits betrafen Android.

web & mobile developer 7/2016 Kaspersky Lab, Security Bulletin 2015/2016

stark in Bewegung und eine Konsolidierung nicht in Sicht. Neben den Branchengrößen wie Dell und Microsoft drängen auch immer mehr Anbieter von CLM-Lösungen (Client Lifecycle Management) auf den Markt für EMM- beziehungsweise MDM-Lösungen – zum Beispiel Baramundi, Landesk und Matrix42. Hinzu kommen zahlreiche Anbieter, die sich auf die Verwaltung mobiler Geräte spezialisiert haben, wie AppTec360 aus der Schweiz oder Cortado und Pretioso aus Deutschland. Auch fast alle Anbieter von Security-Software haben ihr Portfolio um Anwendungen für die Verwaltung mobiler Geräte erweitert.

Das macht die Wahl des richtigen Produkts alles andere als einfach. Interessant sind immer Referenzen. Zum Beispiel kann man sich Case-Studies ansehen und vergleichen, welches Szenario am besten zu den Gegebenheiten im eigenen Unternehmen passt. Man kommt allerdings trotzdem nicht darum herum, einen Anforderungskatalog zu erstellen und sich auf dieser Basis von möglichst vielen Anbietern Lösungen und Angebote erarbeiten zu lassen.

Dass ist die Wahl des Anbieters nicht einfach ist, weiß auch Klaus Düll von Pretioso: »Alle reden von mobilen Konzepten und sind auf Rat angewiesen, den es viel zu wenig gibt. Wie jedes Trendthema lockt auch das Thema Mobility schlechte Berater in Scharen an, die dann Unternehmen aufgrund ihrer Inkompetenz auf falsche Wege schicken.« Häufig wird zum Beispiel der Datenschutz nur halbherzig berücksichtigt: »Viele dieser Berater vergessen zum Beispiel beim Thema BYOD das Thema Datenschutz komplett und bringen die beim Anwender verantwortlichen Führungskräfte hierdurch in eine rechtswidrige Situation.«

Mehr Aufklärung hält auch Sahin Tugcular von AppTec360 für geboten: »Kleine und mittlere Unternehmen beschäftigen sich immer mehr mit dem Thema Enterprise Mobility Management. Hier gibt es größeren Erklärungsbedarf.«

Bei EMM-Lösungen hat man die Wahl zwischen On-Premise-Lösungen und der Cloud. On-Premise-Lösungen kosten meist erst einmal mehr, da sich das Unternehmen um die nötige IT-Infrastruktur selbst kümmern muss. Geld kostet sowohl die Anschaffung als auch die Wartung – dafür liegen die Firmendaten sicher im eigenen Rechenzentrum.

Wer sich für einen Cloud-Dienst entscheidet, der überlässt die Wartung mehr oder weniger dem Anbieter. Das spart in der Regel Geld, da sich zahlreiche Kunden die Infrastruktur in der Cloud teilen. Aber auch die Sicherheit der Server überlässt man dem Anbieter. Angesichts zahlreicher Enthüllungen zu diversen staatlichen Abhöraktionen erfordert eine EMM-Lösung in der Cloud großes Vertrauen in den Anbieter. Aus Datenschutzgründen ist bei einer Cloud-Lösung ein Server-Standort in Deutschland auf jeden Fall zu bevorzugen.

Endgeräte-Infrastruktur

In Unternehmen, in denen Bring Your Own Device umgesetzt wird, ist eine homogene mobile Endgeräte-Infrastruktur meist illusorisch. Als IT-Verantwortlicher sollte man darauf achten, dass alle erlaubten mobilen Plattformen von der EMM-Lösung unterstützt werden. Bei den großen Betriebssystemen wie Android, iOS und Co. ist das ohnehin meist der Fall.

Wenn Mitarbeiter nur vom Unternehmen gestellte Smartphones und Tablets nutzen dürfen, muss man sich als IT-Verantwortlicher aber für ein System entscheiden. Doch welches soll man nehmen? »iOS, da es aktuell über sehr gute Verwaltungsmöglichkeiten verfügt. Ein weiterer Vorteil von iOS ist, dass es nicht so wie bei Android fragmentiert ist und man nicht so viele herstellerspezifische Android-Versionen mit unterschiedlichen Schnittstellen hat«, so die eindeutige Meinung von Sahin Tugcular von AppTec 360.

Ähnlich sieht es Carsten Mickeleit von Cortado: »In Bezug auf den Unternehmenseinsatz hat Apple mit iOS 9 ganz klar die Nase vorn. Google zieht zwar jetzt mit Android for Work nach, liegt allerdings noch etwas hinter Apple zurück. Erfreulich ist aber vor allem, dass beide Plattformen nahezu jeder Sicherheits- und Datenschutzanforderung von Unternehmen gerecht werden, ohne dass komplizierte Third-Party-Lösungen wie SDKs, App-Wrapping oder spezielle E-Mail- oder Container-Clients notwendig sind.« Auch der deutlich höhere Preis von Apple-Geräten sollte kein Hinderungsgrund für deren Einsatz sein: »Die Vorteile für die Produktivität der Mitarbeiter und für die IT-Administration im Unternehmen wiegen den höheren Preis schnell auf«, so Mickeleit weiter.

Kaum eine Rolle spielt Blackberry: Mit dem Ende 2015 vorgestellten Smartphone Priv setzt der Hersteller auf Android. »Blackberry hat seit dem Umstieg auf Android seinen Sinn verloren, da es weit bessere und ausgereiftere Android-Konzepte gibt«, meint Klaus Düll von Pretioso. ■

Konstantin Pfliegl
ist Redakteur bei der Zeitschrift com! professional. Er hat mehr als fünfzehn Jahre Berufserfahrung als Redakteur für verschiedene Print- und Online-Medien.
www.xing.com/profile/Konstantin_Pfliegl

Developer Newsletter

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

web & mobile DEVELOPER Newsletter

Probleme mit der Darstellung | Aktuelles Heft

// news

Stellenbörse für Open Source-Unternehmen
Der Open Source-Branche geht es gut, und mit dem Erfolg wächst der Bedarf an weiteren Mitarbeitern. Dem trägt die Open Source Business Alliance (OSB Alliance) nun Rechnung und startet auf ihrer Website eine Stellenbörse und veröffentlicht in einem ersten Schritt Stellenangebote ihrer Mitgliedsunternehmen.

Add-on-Marktplatz für Node.js-Entwickler
Die Progress-Tochtergesellschaft Modulus hat eine Reihe von Zusatzprodukten auf ihrem Add-on-Marktplatz veröffentlicht. Damit ist es für Node.js-Entwickler einfacher, neue Funktionalitäten schneller in ihre Applikationen einzubauen.

HPI will Benchmarks für Big-Data-Leistungsvergleiche erarbeiten
Das Hasso-Plattner-Institut (HPI) ist Gastgeber des fünften internationalen Workshops zu Leistungsvergleichen im Bereich Big Data, dem so genannten Big Data Benchmarking. Das Treffen, zu dem rund 80 Teilnehmer erwartet werden, findet am 5. und 6. August am HPI in Potsdam statt.

Jetzt kostenlos anmelden:

webundmobile.de

twitter.com/webundmobile

facebook.de/webundmobile

gplus.to/webundmobile

MAGENTO IMAGINE 2016

Event des Jahres

Das wichtigste Magento-Event des Jahres, die Imagine, fand im April in Las Vegas statt.

Zum Auftakt gab es für alle früher Angereisten ein Vorprogramm mit einem Hackathon, Docuthon und inoffiziellen Veranstaltungen wie einem Magerun durch die Wüste. Zum eigentlichen Event kamen dann unter dem Motto »We are Magento« mehr als 2500 E-Commerce-Experten von annähernd 400 Agenturen, Partner- und Kundenunternehmen aus mehr als 45 Ländern in Las Vegas zusammen (Bild 1). Anders als bei früheren Veranstaltungen, die am ersten Tag eher ruhig einstiegen und Zeit für Networking boten, ging es diesmal gleich zur Sache. Während einige Community-Entwickler sich noch fragen, ob Magento mit der Version 2.0 nicht etwas zu »enterprisey« geworden ist, und sich einfachere Installationsmöglichkeiten wünschen, ging es in den Magento U Vorträgen (U steht für University) um businessorientierte Themen, beispielsweise darum, wie das Requirements Engineering professionalisiert werden kann.

2500 E-Commerce-Experten nahmen am wichtigsten Magento-Event des Jahres teil (Bild 1)

Mehr in Form einer Schulung denn als Vortrag wurden von Steve Kukla, Lead Trainer und Certified Solution Specialist bei Magento, effektive Strategien und Best Practices (zum Beispiel User-Stories) beschrieben, mit denen sich Anforderungen zielorientiert erheben lassen, um Magento-Projekte qualitativ hochwertig und erfolgreich zu realisieren. Für große Agenturen und erfahrene Business-Solution-Spezialisten ist dies nicht wirklich neu, es zeigt aber, dass Magento professionelle Standards setzt und einen erhöhten Anspruch an alle Magento-Lösungserbringer stellt.

In den Magento 2 Developer Deep Dive Sessions wurde näher auf technische Aspekte in Magento 2 eingegangen. Es gab einen Ausblick auf die nächsten Schritte im Bereich der Frontend-Entwicklung, die sich um alternative Präprozessoren und Stylesheet-Sprachen (neben LESS zum Beispiel Sass oder Stylus) drehen, aber auch zum Backend, wo Asynchronous Contract Call Patterns eingeführt werden und damit die Service Contracts erweitert werden.

Der neue Marketplace

Am zweiten Tag lag der Fokus dann stärker auf der E-Commerce-Software und dem Ökosystem. Mittlerweile wurde Magento 2 schon 228.000-mal heruntergeladen. Es sind 800 Shops online und mehr als 105 Partner haben sich bisher für die Anforderungen der neuen Software schulen lassen.

Als erste große Neuigkeit des Tages wurde dann der Magento Marketplace vorgestellt. Magento geht hier selbstbewusst neue, innovative Wege und vermarktet alle Extensions ab sofort selbst. Dabei will Magento besonders auf Qualität und Integrierbarkeit achten und als erster Ansprechpartner gegenüber den Kunden dienen. Die Abrechnung kostenpflichtiger Extensions läuft zukünftig direkt über Magento, die Entwickler erhalten ihren Anteil überwiesen.

Mit der Zentralisierung und Professionalisierung des Marktplatzes schützt Magento Commerce seine hohen Technologie- und Service-Standards, die es seit dem Start in die Eigenständigkeit auszeichnen. Auch wenn Erweiterungen über Extensions aus dem Marketplace für professionell betreute Magento-Shops grundsätzlich nicht das ideale Vorgehen sind – Erweiterungen sollten immer sauber über Versionierungssysteme ausgerollt werden –, ist der Marketplace ein weiterer Schritt nach vorn. Dabei sind vor allem das Bewertungs-System und die Anzeigen zur Verbreitung der Bewertung interessant, um die Nützlichkeit und Stabilität einer Extension einzuschätzen.

Ankündigung der Magento 2.1

Zu den wichtigsten Neuerungen der Version 2.1 zählt das Staging System, das im Magento Backend übersichtlich und benutzerfreundlich verschiedene Instanzen für Content- und Produkt-Staging ermöglicht. Die native Integration von Elasticsearch wird als weitere Neuerung mit 2.1 eine deutlich flexiblere und performantere Suche ermöglichen, als man es von der Magento-Standardsuche gewohnt ist. Beide Neuerungen richten sich insbesondere an professionelle E-Commerce-Betreiber, die dank Staging-Systemen agil und parallel an Verbesserungen an der eigenen Plattform arbeiten wollen. Außerdem wird die neue Version RabbitMQ bieten.

Weitere Neuigkeiten betreffen das Magento Commerce Order Management. Als zentrale Schnittstelle zwischen Verkauf und Logistik soll die Lösung so ziemlich alles bieten, um einen nahtlosen Omnichannel zu realisieren. Dabei richtet sich die neue Leistung an diejenigen Kunden, deren Prozessen eine gewisse Komplexität aufweisen und die nicht jeden Workflow des Order- und Bestandsmanagements in ihr ERP-System einbauen wollen.

Cloud Edition für Enterprise-Kunden: Alles aus einer Hand (Bild 2)

Magento adressiert hier insbesondere Händler, die verteilte Logistik-Infrastrukturen abbilden müssen, also mehrere Lager, Stores und Dienstleister managen wollen. Das Wertschöpfungspotenzial nimmt dabei mit der Komplexität des Commerce-Modells und des Bestellvolumens zu. Mittlerweile setzen schon sieben Unternehmen das neue Magento Commerce Order Management ein, darunter auch Fraport.

Cloud Edition für Enterprise-Kunden

Das beherrschende Thema des zweiten Tages war die Ankündigung einer Magento Enterprise Cloud Edition. Magento hat dafür eine moderne Cloud-Infrastrukturlösung geschaffen. Unter Verwendung von platform.sh, New Relic, Blackfire.io und den Amazon Web Services (AWS) wird eine PaaS-Umgebung (Platform as a Service) bereitgestellt. Magento spricht in diesem Kontext auch von einer Digital Commerce Plattform. In ihr sollen E-Commerce-Lösungen hochgradig skalierbar aufgebaut werden können.

Dabei geht Magento weiter, als einfach nur virtuelle Server zur Installation anzubieten, und nutzt das gesamte Spektrum der verteilten AWS-Dienste zur intelligenten Skalierung. Magento bietet dann alles aus einer Hand: Bereitstellung, globale Verfügbarkeit, performante Auslieferung, Stabilität, dynamische Skalierbarkeit, Release-, Patch- und Cloud-Management sowie umfassendes Monitoring und Support. Steve Yankovich, Chief Product Officer von Magento Commerce, nannte die Magento Enterprise Cloud Edition daher auch zu Recht einen Gamechanger (Bild 2).

Für die Magento Enterprise Cloud Edition wird es ein neues Lizenzmodell geben. Es wurde bereits bekannt gegeben, was die Lizenz umfassen soll: Nutzung eines Content Delivery Networks, die immer neuesten Versionen von New Relic und Blackfire, die Magento-Enterprise-Lizenz selbst, die Bereitstellung von Entwickler-Tools auf einer skalierbaren Plattform ohne starre Beschränkung mit bis zu acht Staging-Umgebungen und einer dreifach redundanten, dedizierten Produktivumgebung, abgesichert durch einen 24/7-Support. Nur der Preis ist zum aktuellen Zeitpunkt noch nicht bekannt.

In einem ergänzenden Vortrag gingen Magento und Amazon detaillierter auf die AWS-Leistungen ein und machten deutlich, dass sie besonderen Wert auf die Einhaltung des Datenschutzes nach EU-Richtlinien legen. Für den deutschen Markt werden die Amazon Web Services über Frankfurt angeboten. Damit wäre die Magento Enterprise Cloud Edition auch für deutsche Firmen eine hochinteressante Alternative. Und im Gegensatz zu Magento Go haben Entwickler hier den vollen Zugriff auf den Quellcode, können Magento Enterprise wie gewohnt umfassend für Kundenanforderungen anpassen und haben, wie es scheint, auch die Möglichkeit, die von Magento bereitgestellten Entwickler-Tools zu ersetzen.

Unter all den Neuigkeiten des letzten Veranstaltungstags stach eine ganz besonders heraus: Magento hat eine Magento Enterprise B2B Edition angekündigt. Nach aktuellen Zahlen (Internet Retailer B2B eCommerce Top 300) ist das Unternehmen schon jetzt weltweit der größte Player im B2B E-Commerce, noch vor SAP/hybris, IBM und Netsuite.

Mit der neuen Edition, die über speziell an die Bedürfnisse von B2B-Kunden angepasste Funktionen wie Nutzer- und Rollenmanagement, kundenindividuelle Produkt- und Preislisten und einem Quote-Workflow für individuelle Angebots- und Bestellabwicklungen verfügt, wird sich der Abstand wohl noch vergrößern. Als Magento B2B-Partner können wir zudem eines schon versprechen: Es werden in den kommenden Monaten noch viele weitere relevante Funktionen für den Geschäftskundenbereich hinzukommen.

Fazit

Zieht man ein Fazit der diesjährigen Magento Imagine, kann dies aus Agentursicht nur positiv ausfallen. Magento Commerce hat die Wünsche der Agenturen fast eins zu eins umgesetzt und mit einer wachsenden Feature-Liste, der Enterprise Edition, dem Technologieupdate und der nun auch nativ unterstützten B2B-Edition gezeigt, dass man die Dynamik im Markt wirklich beherrscht. Der Wechsel von Ebay zu einem freien Unternehmen mit Rückendeckung des Investors Permira hat Magento rechtzeitig den einzigartigen Speed und die herausragende Innovationskraft zurückgegeben, die es zum Weltmarktführer gemacht haben. ■

Ralf Lieser
hat viele Jahre Erfahrung in der Software-Entwicklung sowie Integration von E-Commerce-Plattformen auf Basis von Magento. Er überwacht als Leiter des Qualitätsmanagements die Umsetzung von E-Commerce-Projekten.
www.netz98.de

CROSS-BORDER-HANDEL

Grenzüberschreitend

Grenzüberschreitende Geschäfte stellen Händler vor juristische Probleme.

Das World Wide Web kennt – wie der Name schon erkennen lässt – keine Ländergrenzen. Das stellt gerade Betreiber von Online-Shops vor große Herausforderungen. Oftmals machen sich Online-Händler keine oder nur sehr wenig Gedanken dazu, wer die eigenen Waren beziehungsweise Dienstleistungen in Anspruch nehmen soll, sprich: wie die eigene Kundenzielgruppe im Detail aussieht. Bei der Bestimmung dieser Zielgruppe gibt es diverse Kriterien zu beachten; ein ganz entscheidendes ist die Nationalität.

Will man nämlich auch an ausländische Kunden verkaufen, oder aber vielleicht doch nur innerhalb Deutschlands, aber der eigene Webshop ist so gestaltet, dass er auch auf internationale Kunden abzielt, dann kann es unter Umständen sein, dass auch Rechtsordnungen fremder Länder zu beachten sind (Bild 1).

Bei fehlender Kenntnis ist das natürlich gar nicht erst möglich, aber auch bei vorhandener Kenntnis dieses Umstands sind die zu bewältigenden Aufgaben nicht gerade gering. Zwar ist das europäische Verbraucherschutzrecht weitgehend harmonisiert, es herrscht also in allen Mitgliedsländern der Europäischen Union ein sehr ähnliches Niveau der einschlägigen Rechtsvorschriften (Bild 2). Allerdings steckt der Teufel oftmals im Detail, sodass auch die kleinsten Abweichungen vom deutschen Rechtssystem zu großen Auswirkungen in der Praxis führen können.

Außerdem gibt es ja auch noch die Schweiz, die als deutschsprachiges Land insofern eine gewisse Sonderstellung einnimmt, als sie gerade kein EU-Mitglied ist, dort aber für viele deutsche Webshop-Betreiber ein interessantes Kundenpotenzial besteht.

Salesupply unterstützt Online-Händler bei deren Internationalisierungsplänen (Bild 1)

Nachfolgend werden drei exemplarische Situationen skizziert, um das Ausmaß zu verdeutlichen.

Deutscher Shop und Schweizer Kunde: Eine Schweizer Privatperson kauft Waren in einem deutschen Online-Shop. In den AGB hat der Shop-Betreiber unter anderem die Klauseln aufgenommen: »Lieferung in die DACH-Region (Deutschland, Österreich und Schweiz)«, »14-tägiges Widerrufsrecht für Verbraucher« und »Es gilt deutsches Recht«. Im Grunde nichts Ungewöhnliches, diese oder ähnliche Klauseln finden sich standardmäßig in zahlreichen Shop-AGB.

In diesem Beispiel widerruft der Schweizer Kunde fristgerecht den Kaufvertrag. Anders als in den EU-Mitgliedsstaaten gibt es in der Schweiz selbst kein Widerrufsrecht bei Online-Käufen. Hier kann der Schweizer Kunde ausnahmsweise doch widerrufen, weil der Shop-Betreiber dies durch die Gestaltung seiner AGB freiwillig auch Schweizer Kunden eingeräumt hat.

In einem anderen Fall handelt es sich um einen deutschen Shop und einen spanischen Kunden. Der spanische Verbraucher bestellt Waren in einem deutschen Webshop, dessen Betreiber auch eine Niederlassung in Spanien hat. Die Waren werden geliefert, der Kaufpreis bezahlt, zunächst gibt es keine Probleme.

Einige Zeit später erhält der Händler ein Schreiben der spanischen Datenschutzbehörde Agencia Española de Protección de Datos (AEPD) mit der Aufforderung, bestimmte Auskünfte zu erteilen und eine Geldbuße zu zahlen.

Dazu muss man wissen, dass spanische Online-Händler sich vor Aufnahme ihrer Tätigkeit bei der AEPD registrieren lassen müssen und zudem den teilweise recht strengen Datenschutzregelungen ihres Landes unterliegen. Dies gilt etwa in Bezug auf den Einsatz von Cookies oder den Versand von Newslettern. Für deutsche Online-Händler kann also das spanische Datenschutzrecht dann zur Anwendung kommen, wenn sie eine Niederlassung vor Ort betreiben und sich dennoch nicht bei der AEPD registrieren.

Handel mit Italien

Ähnliches kann deutschen Online-Händlern übrigens passieren, wenn sie Verträge mit italienischen Kunden abschließen. In Italien muss man sich eventuell mit der Wettbewerbsbehörde auseinandersetzen. Diese kann nämlich auch gegenüber ausländischen Unternehmen Sanktionen und insbesondere Geldbußen verhängen, wenn sie Verstöße gegen italienisches Verbraucherrecht feststellt.

Das europäische Verbraucherschutzrecht ist weitgehend harmonisiert (Bild 2)

In einem weiteren Beispiel geht es um einen deutschen Shop und einen französischen Kunden: Ein Verbraucher aus Frankreich kauft Waren in einem deutschen Webshop. Zum Zeitpunkt der Bestellung war die Ware versehentlich falsch ausgezeichnet, statt mit 100 Euro war der Preis mit 10 Euro angegeben.

Der Shop-Betreiber meldet sich sogleich bei seinem französischen Kunden, dieser besteht jedoch auf die Erfüllung des Vertrags. Vor dem Hintergrund, dass in Frankreich – im Unterschied zu Deutschland und vielen anderen EU-Staaten – die Warenpräsentation im Online-Shop als verbindliches Vertragsangebot angesehen wird, ist das für den deutschen Händler natürlich ein Problem. Denn das bedeutet, dass französische Kunden mit Abgabe ihrer Bestellung einen verbindlichen Vertrag abschließen.

In Deutschland kommt es erst dann zu einem verbindlichen Vertragsabschluss, wenn der Händler die Annahme des Vertrags ausdrücklich per E-Mail oder stillschweigend durch den Versand der Waren an den Kunden erklärt. In diesem Beispiel ist die Rechtslage so zu bewerten, dass der Vertrag mit dem französischen Kunden bereits mit dessen Bestellung als geschlossen gilt. In Bezug auf die fehlerhafte Preisangabe hat der Händler somit keinen Spielraum mehr, er hat nur noch die Möglichkeit, den Vertrag wegen Irrtums anzufechten, was allerdings wohl schwierig zu realisieren sein wird.

Kriterien

Wie diese Beispiele plastisch belegen, gibt es diverse Fallen, in die man tappen kann. Die entscheidende Frage lautet also: Wann zielt ein deutscher Webshop auf ausländische Kunden, wann handelt es sich also um einen sogenannten Cross-Border-Shop?

Das lässt sich anhand der folgenden Kriterien beantworten:

- ausdrücklicher Hinweis auf ausländische Kunden beziehungsweise einen ausländischen Markt,
- verschiedene Sprachversionen, ▶

developer media

EPUB oder PDF je 9,99 €

Vorankündigungen

.NET WPF
Moderne Desktop-Anwendungen mit .NET
04 2016

ASP.NET
Microsofts neuer Weg der Webentwicklung
05 2016

Data Science
Intelligente Verarbeitung großer Datenmengen
06 2016

.NET-Architektur
So wird aus Anforderungen eine perfekte Anwendung
07 2016

Flexible Projektstrukturen für .NET
08 2016

Angular 2
Modulares Frontend für den Browser
09 2016

Digitale Revolution in der IT
11 2016

Weitere devbooks unter
www.developer-media.de/devbooks

- Verweis auf ausländische Rechtsvorschriften,
- mehrsprachige Rechtstexte (AGB, Impressum, Datenschutzerklärung …),
- Nutzung der Top-Level-Domain eines bestimmten Landes beziehungsweise unterschiedlicher Länder,
- gezielter Auslandsversand beziehungsweise die Möglichkeit zur Abholung der Ware im Ausland,
- Akzeptanz ausländischer Währung,
- Angabe einer ausländischen Bankverbindung,
- ausländische Telefonnummer / Service-Hotline,
- Art beziehungsweise Einzugsgebiet des Anbieters,
- Werbung auf ausländischen Internetseiten.

Je mehr der hier aufgeführten Kriterien auf einen Webshop zutreffen, desto eher ist von einer Cross-Border-Ausrichtung auszugehen. Das wiederum hat zur Folge, dass der Betreiber sich auch mit den unterschiedlichen Rechtslagen der Staaten beschäftigen muss, auf die er abzielt. Hierzu genügt es definitiv nicht, die deutschen Rechtstexte, also die AGB, die Widerrufsbelehrung, das Impressum oder die Datenschutzerklärung, einfach eins zu eins in andere Sprachen übersetzen zu lassen. Es müssen vielmehr die jeweiligen ausländischen Rechtsordnungen in die inhaltliche Gestaltung der fremdsprachigen Rechtstexte einfließen. Sofern man im Einzelfall ausländischen Verbrauchern stärkere Rechte zugesteht, als ihnen nach ihrer nationalen Rechtsordnung eigentlich zustünden, dann ist das kein Problem, sondern unter Umständen vielleicht etwas lästig für den Händler. Entzieht man ausländischen Kunden allerdings bestimmte Rechtspositionen, dann kann dies hierzulande einen Verstoß gegen das (deutsche) Wettbewerbsrecht darstellen und entsprechend abgemahnt werden.

Rechtsprechung

Auf der anderen Seite haben sich Betreiber ausländischer Webshops auch an die Vorgaben des deutschen Gesetzgebers zu halten, wenn sie Verträge mit deutschen Verbrauchern abschließen.

Für den Bereich der Anbieterkennzeichnung hat das Landgericht (LG) Stuttgart in seinem Beschluss vom 13. Dezember 2011 (Aktenzeichen: 17 O 408/11) entschieden, dass ein ausländischer Betreiber einer gewerblich ausgerichteten Internetseite, die sich an deutsches Publikum wendet, auch an die deutschen Gesetze zu halten hat. Folglich, und das ist inzwischen unstrittig, müssen auch ausländische Internetpräsenzen mit Zielrichtung deutscher Verbraucher ein formell und inhaltlich korrektes Impressum bereitstellen.

Das LG Karlsruhe gelangt in seinem Urteil vom 16. Dezember 2011 (Aktenzeichen: 14 O 27/11 KFH III) zu der Erkenntnis, dass das in Deutschland bei Fernabsatzgeschäften zu beachtende Widerrufsrecht zwingendes Verbraucherschutzrecht darstellt. Da es außerdem auf der sogenannten EU-Fernabsatzrichtlinie basiert, kommt das Widerrufsrecht auch bei vielen innerhalb des Gebietes der Europäischen Union abgeschlossenen Verträgen zur Anwendung. Das gilt jedenfalls im Verhältnis Unternehmer–Verbraucher, also im Bereich Business-to-Consumer (B2C). Online-Händler haben grundsätzlich die Möglichkeit, in ihren AGB eine Rechtswahlklausel zu integrieren. Dadurch können sie das Land festlegen, dessen Rechtsordnung anwendbar sein soll. Diese zulässige Rechtswahlmöglichkeit erfährt insoweit eine Einschränkung, als dadurch privaten Kunden keine für sie anwendbaren Verbraucherrechte abgeschnitten werden dürfen. Daher ist auch immer das nationale Verbraucherrecht desjenigen Staates zu beachten, in dem der betreffende Kunde seinen Wohn- beziehungsweise Aufenthaltsort hat. Deutschen Kunden steht folglich bei Verträgen mit ausländischen Shop-Betreibern auch zwingend deutsches Recht zur Seite.

Verbraucher mit Sitz in der EU haben die Möglichkeit, bei Rechtsstreitigkeiten mit im europäischen Ausland sitzenden Online-Händlern diese auch in ihrem Heimatland zu verklagen. Deutsche Endkunden können also beispielsweise einen polnischen Händler vor einem deutschen Gericht verklagen. Dies hat jedenfalls der Europäische Gerichtshof (EuGH) in seinem Urteil vom 6. September 2012 (Aktenzeichen: C-190/11) so entschieden. Dies bringt natürlich einen Heimvorteil für Verbraucher mit sich.

■

Problemfall Versandkosten

Sollen die eigenen Waren auch an ausländische Kunden versandt werden, so sind die Versandkosten in einem ohnehin bereitzustellenden Menüpunkt so detailliert wie möglich aufzuführen. Angaben, wie etwa »Kosten für den Versand außerhalb Deutschlands nur auf Anfrage« sind tabu. Sind genaue Preisangaben nicht möglich, so ist zumindest die Berechnungsgrundlage aufzuführen.

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

Michael Rohrlich
ist Rechtsanwalt und Fachautor aus Würselen. Seine beruflichen Schwerpunkte liegen auf dem Gebiet des Online-Rechts und des gewerblichen Rechtsschutzes.

www.ra-rohrlich.de

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. Günther Götz

Chefredakteur
Max Bold
– verantwortlich für den redaktionellen Teil –
E-Mail: redaktion@webundmobile.de

Schlussredaktion
Ernst Altmannshofer

Redaktionelle Mitarbeit
Philip Ackermann, Johann Baumeister, Christian Bleske, Tam Hanna, Anna Kobylinska, Bernhard Lauer, Ralf Lieser, Patrick Lobacher, Filipe Martins, Florence Maurice, Konstantin Pfliegl, Frank Pientka, Holger Reibold, Michael Rohrlich, Markus Schraudolph, Katharina Sckommodau, Thomas Sillmann, Frank Simon

Art Directorin
Maria-Luise Sailer

Grafik & Bildredaktion
Alfred Agatz, Dagmar Breitenbauch, Verena Greimel, Hedi Hefele, Manuela Keller, Simone Köhnke, Cornelia Pflanzer, Karoly Pokuta, Petra Reichenspurner, Ilka Rüther, Sebastian Scharnagl, Christian Schumacher, Nicole Üblacker, 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

© 2016 Neue Mediengesellschaft Ulm mbH

Jetzt Ihre web & mobile developer auf dem iPad lesen

developer media
Webinare

Jetzt online weiterbilden!

„Fortschritt heißt für mich vor allem, dass man fortschreiten will."

Johannes Hoppe
IT-Berater, Programmierer, Webdesigner

developer-media.de/webinare

ARBEITSMARKT

TRENDS UND JOBS FÜR ENTWICKLER

Monatliches Ranking

Zur Zukunft der Arbeit

Die Digitalisierung von Wirtschaft und Industrie ist in vollem Gange, und damit nehmen auch die Ängste vor einem Job-Verlust zu. Intensiv mit diesem Thema beschäftigt sich Sven Gabor Janszky, Direktor des Zukunftsforschungsinstituts 2b AHEAD ThinkTank. Er prognostiziert vor dem Jahr 2040 eine Phase der Vollbeschäftigung. Erst nach 2040 ändern sich dann die Vorzeichen und Wissensarbeiter werden von Computern ersetzt. Ausgeglichen werden soll die entstehende Job-Lücke durch neue identitätsstiftende Jobs, Tätigkeiten, die wir heute noch nicht kennen, beziehungsweise von denen wir heute nicht vermuten würden, dass man damit Geld verdienen kann.

Die monatliche Auswertung der Jobangebote für Webentwickler anhand der Datenbank der Metasuchmaschine Jobkralle.de ergab, dass die meisten Stellenangebote Mitarbeiter in Bayern, NRW und Baden-Württemberg suchen. Im Ländervergleich landeten Berlin und Hessen auf den Plätzen 4 und 5. Im Vergleich der Großstädte (Bild 1) waren München, Berlin und Hamburg ganz oben zu finden, die zusammen knapp die Hälfte (48,9 Prozent) der Angebote auf sich vereinen. Auf einen ebenfalls noch zweistelligen Anteil kam Frankfurt am Main.

Dieselbe Städte-Reihenfolge bei sehr ähnlichen Prozentzahlen gilt auch für die Jobangebote für Mobile-Entwickler. Allerdings liefert die Jobdatenbank für die Stichworte »Mobile-Entwickler« und »Mobile-Developer« insgesamt nur rund halb so viele Treffer wie für die Stichwörter »Web-Entwickler« und »Web-Developer«. Das Ergebnis des Bundesländervergleichs zeigt Bild 2. Auch hier ähneln die Ergebnisse denen für Webentwickler sehr stark. Allerdings sind die prozentualen Anteile von Berlin und Hamburg in dieser Erhebung größer als im Ländervergleich der Jobs für Webentwickler.

Das wichtigste Technologie-Thema in aktuellen Stellenanzeigen bleibt das Thema Cloud (Tabelle 1). Das Stichwort »Big Data« tauchte Ende April nicht mehr ganz so stark in Stellenanzeigen auf wie noch im Monat zuvor. Dagegen legte die Nachfrage nach SharePoint-Kenntnissen kräftig zu. Deutlich zurück geht die Trefferzahl beim Stichwort »Responsive Web«. Womöglich deshalb, weil diese Anforderung an Webprogrammierer langsam, aber sicher zum Standard wird, der nicht mehr extra erwähnt wird.

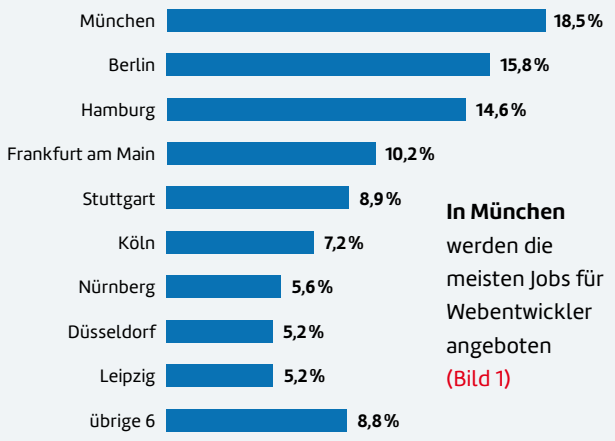
Ein Blick auf die Nennungen von Programmiersprachen in den Titelzeilen von Jobangeboten zeigte, dass Java immer noch mit Abstand am häufigsten genannt wird (43,8 Prozent), gefolgt von PHP (15,9 Prozent), C# (11,9 Prozent) und C++ (8,0 Prozent). JavaScript kommt mit 5,3 Prozent auf den sechsten Rang.

Tabelle 1

Rang	Technologie	Anteil *
1	Cloud	18,5 %
2	MySQL	12,6 %
3	HTML5	10,7 %
4	SharePoint	8,0 %
5	Big Data	7,2 %
6	Android	6,6 %
7	iOS	6,4 %
8	Microsoft SQL Server	6,4 %
9	CSS3	5,2 %
10	Angular.js	3,8 %
11	Windows 10	3,2 %
12	WPF	2,9 %
13	Responsive Web	2,7 %
14	NoSQL	2,6 %
15	Azure	1,8 %
16	WCF	1,5 %

* Prozentualer Anteil der Treffer

Jobs für Webentwickler



In München werden die meisten Jobs für Webentwickler angeboten (Bild 1)

web & mobile developer 7/2016 Eigene Erhebungen, Jobkralle.de

Jobs für Mobile-Entwickler

Für Jobs in Bayern und NRW sind derzeit die meisten Stellenanzeigen für Mobile-Entwickler zu finden (Bild 2)

web & mobile developer 7/2016 Eigene Erhebungen, Jobkralle.de

Zahl des Monats

Leicht angestiegen auf insgesamt **35.439** ist nach Angaben des Statistischen Bundesamts die Zahl der Erstsemester im Informatikstudium. Darunter waren **7868** Frauen, der Frauenanteil beträgt damit rund **25 Prozent**. Das ist deutlich mehr als in anderen technischen Studiengängen (Maschinenbau: 12 Prozent. Elektrotechnik: 14 Prozent).

Quelle: Statistisches Bundesamt, Bitkom

Weltweite Studie

Entwickler sind Autodidakten

Ein großer Teil deutschen Entwickler sind Autodidakten, das hat die weltweite Umfrage von Stack Overflow ergeben. 74,7 Prozent der Befragten gaben an, zumindest einen Teil ihrer Fähigkeiten selbst erlernt zu haben (Bild 3). Für Unternehmen gilt deshalb: Wer bei der Beurteilung von Bewerbungsunterlagen zunächst auf den Universitätsabschluss achtet, verpasst die besten Bewerber. 42,5 Prozent nutzen zur Fortbildung berufliche Weiterbildungen. Dagegen sind Online-Kurse wie etwa Coursera oder Codeacademy mit 19,8 Prozent noch wenig verbreitet (weltweit 26,18 Prozent). 26,7 Prozent der Entwickler in Deutschland haben einen Master-Abschluss, weltweit sind es lediglich 18,6 Prozent. In Sachen Verdienstmöglichkeiten liegen die USA (Schnitt: 83.080 Euro) und Großbritannien (58.110 Euro) vorne. Trotz des hohen Abschlussniveaus kommen deutsche Entwickler nur auf durchschnittlich 46.940 Euro Jahressalär.

www.stackoverflow.com

Bitkom

IT-Studium nur für Frauen

In Deutschland richten sich von insgesamt 1600 IT-nahen Studiengängen sechs speziell und ausschließlich an Frauen. »Frauenstudiengänge stellen die gleichen fachlichen Anforderungen wie sonstige Studiengänge, aber sie verändern die Spielregeln in den Lernteams und senken die Einstiegshürden für Frauen in die IT. Auch weil sie oftmals Rücksicht auf die spezifischen Bedürfnisse von Studentinnen nehmen, die familiäre Verantwortung haben, indem beispielsweise Pflichtveranstaltungen auf Kita-Öffnungszeiten abgestimmt werden«, sagte Bitkom-Präsidiumsmitglied Martina Koederitz anlässlich des Girls' Day. Frauenstudiengänge bieten die Hochschule für Technik und Wirtschaft Berlin, die Hochschule Bremen, die Hochschule Furtwangen, die Jade-Hochschule Wilhelmshaven, die Fachhochschule Stralsund sowie die Ernst-Abbe-Hochschule Jena an.

www.bitkom.org

StepStone

Wegweiser für die Gehaltsverhandlung

Jede vierte Fachkraft in Deutschland hat ihren Chef noch nie nach einer Gehaltserhöhung gefragt. Dabei lohnt es sich, wie Ergebnisse des StepStone Gehaltsreports 2016 zeigen: Wer das Thema Gehalt aktiv anspricht, verdient im Schnitt 21 Prozent mehr. So holen Sie das Beste für sich raus (Bild 4):

Ordnen Sie Ihre Gehaltsvorstellungen ein. Das Wissen um übliche Gehälter in Ihrem Berufsfeld und Ihrer Branche ist von essenzieller Bedeutung für die Gehaltsverhandlung. Eine Orientierung bietet der StepStone Gehaltsreport (www.stepstone.de/gehaltsreport).

Wählen Sie den richtigen Ansprechpartner. In der Regel ist der direkte Vorgesetzte der passende Ansprechpartner.

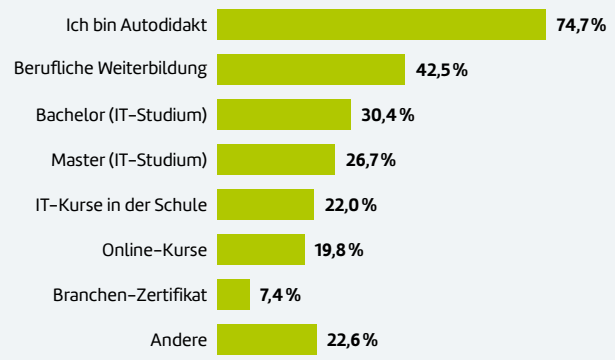
Bestimmen Sie einen passenden Zeitpunkt. Jede dritte Fachkraft, die aktiv nach einer Gehaltserhöhung fragt, tut das einmal jährlich.

Bereiten Sie den Termin gezielt vor. Bitten Sie Ihren Chef mit ausreichendem Vorlauf vor der eigentlichen Gehaltsverhandlung um ein Feedbackgespräch, in dem Sie über Ihren Beitrag zum Unternehmenserfolg sprechen.

Achten Sie auf Formulierungen. Dann vereinbaren Sie mit etwas Abstand einen neuen Termin, etwa für eine Besprechung zu einem Projekt oder zur strategischen Ausrichtung Ihrer Position.

www.stepstone.de

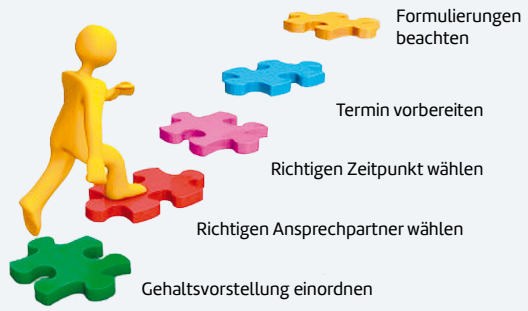
Bewerbungen verbessern



Drei Viertel der deutschen Entwickler gaben an, zumindest einen Teil ihrer Fähigkeiten selbst erlernt zu haben (Bild 3)

web & mobile developer 7/2016 StackOverflow, Mehrfachnennungen möglich

Wie Entwickler ihren Job finden



Thema Gehalt: Wer es aktiv anspricht, verdient mehr. Diese fünf Schritte führen zum Erfolg (Bild 4)

web & mobile developer 7/2016 Quelle: StepStone

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

prodot – People. Passion. Performance.
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. prodot bringt hierzu die richtigen Menschen zusammen.
Seit über 15 Jahren vertrauen uns deshalb Marktführer wie Aldi Süd, Microsoft und Siemens.
Sprechen Sie mich an – als Kunde, Partner oder Kollege. Pascal Kremmers.

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.

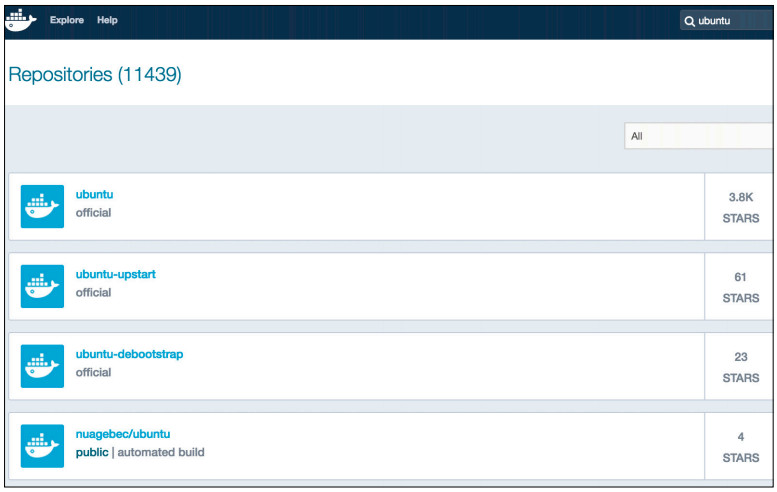
Web- / Mobile-Entwicklung & Content Management

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 8/2016 erscheint am 14. Juli 2016

Container-Virtualisierung



Ob virtualisierter Server, Windows-Betriebssystem auf dem Mac oder ein Ubuntu in der VirtualBox, als Entwickler kommt man früher oder später mit dem Thema Virtualisierung in Kontakt. Das in der Programmiersprache Go geschriebene Docker ist auch ein Produkt zur Virtualisierung, nämlich zur sogenannten Container-Virtualisierung. Systeme, die virtualisieren, werden als Hypervisor bezeichnet. Hypervisoren werden abhängig vom Grad der Virtualisierung in unterschiedliche Kategorien eingeteilt. Typ-1-Hypervisioren abstrahieren direkt die Hardware, Systeme zur Container-Virtualisierung sind grundsätzlich Typ-2-Hypervisoren. Im Gegensatz zu Systemen, die das ganze Betriebssystem virtualisieren, wird bei Container-Virtualisierung der Linux-Kernel untereinander geteilt.

Eigene Frameworks für iOS

Als iOS-Entwickler hat man es ständig mit Frameworks zu tun. Sei es nun das Foundation-, das UIkit- oder das Core-Data-Framework; Apple bietet eine Vielzahl an vorgefertigten Frameworks für die iOS-Entwicklung an, von denen manche – wie UIkit – unabdingbar für die tägliche Arbeit sind. Diese Frameworks liefern ganz grundlegende Klassen und Funktionen, die sich quer durch die verschiedensten iOS-Projekte hinweg verwenden lassen.

Promise-Programmiermuster

Asynchrone Programmiermuster mit CPS und der damit einhergehenden Inversion of Control führen zu unübersichtlicherem Code und erschwerter Kombinierbarkeit von Programmbausteinen. Promises sind eine Lösung für diese Probleme. Die Idee dahinter ist einfach: Vor der Inversion of Control war die Welt noch in Ordnung, also kann man versuchen, wieder die einfache Weltsicht synchroner Funktionsaufrufe herzustellen.

UX-Design und Scrum

In der Software-Entwicklung setzt sich Scrum als agile Projektmanagement-Methode zunehmend durch. Im Scrum-Kontext wurde es bisher jedoch vernachlässigt, andere Elemente und fachliche Disziplinen, die essenziell für ein starkes Produkt sind, zu integrieren – wie beispielsweise das User Experience Design. Eine gute User Experience (UX) führt dazu, dass Anwender ein Produkt beziehungsweise eine Software gerne nutzen.

dotnetpro

Ausgabe 7/2016 ab 16. Juni 2016 am Kiosk

Am Anfang war das Wort. Dieses Zitat umschreibt in exzellenter Weise das Schwerpunktthema der nächsten dotnetpro. Es geht dabei um domänenspezifische Sprachen.

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