Canvas und SVG: Zeichnen in HTML5 S. 40

Flexible Websites mit Element Queries S. 64

PHP: Statische Code-Analyse mit Phan S. 116

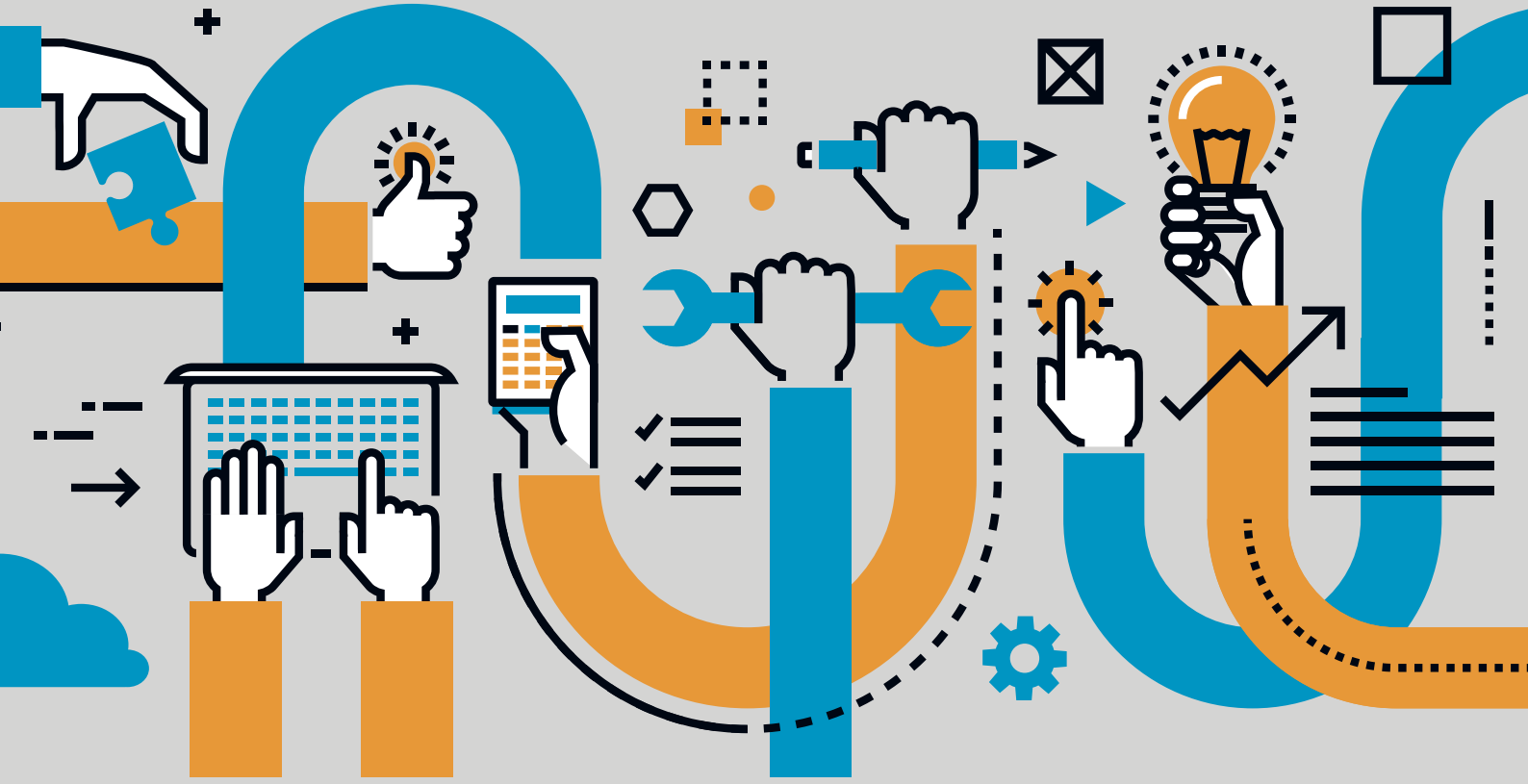
web & mobile DEVELOPER

webundmobile.de

web & mobile DEVELOPER

Bessere Websites mit WebComponents

Die Vorteile der komponentenorientierten Webentwicklung. S. 22



UserNotifications-Framework

Thomas Sillmann stellt das von Apple ab iOS 10 neu eingeführte Framework im Detail vor S. 70

Ausgabe 11/16

Deutschland 14,95 EUR

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

4 198255 014951 11

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

Komponenten

Diese Ausgabe befasst sich in einem Schwerpunktthema mit der komponentenorientierten Webentwicklung, die eine immer größere Rolle spielt.

Als Webentwickler hört man häufiger denn je den Begriff Komponente. Sei es beim Thema WebComponents, in dem viele die Zukunft des Webs sehen, in React Components oder neuerdings auch Angular 2 Components. Was hat es damit auf sich, und warum ist das wichtig? Welche Möglichkeiten haben Entwickler heute, mit Komponenten zu entwickeln? Diese Fragen beantwortet Jochen Schmidt im Schwerpunktthema dieser Ausgabe (ab Seite 22). Außerdem stellt er in einem weiteren Artikel die Bibliothek Preact vor. Es handelt sich dabei um den Versuch, die Kernvorteile von React mit so wenig Code wie möglich nachzubauen, damit es in bestehenden Anwendungen oder bei sehr knappen Ressourcen ohne Probleme integriert werden kann (ab Seite 32).

»Welche Möglichkeiten haben Entwickler heute, mit Komponenten zu entwickeln?«

Wenn man beim Design einer Website mit den klassischen, Viewport-basierten Media Queries nicht weiterkommt, schlägt die Stunde der Element Queries. Was tut man beispielsweise, wenn bei einem Projekt die Tabelle auf manchen Unterseiten im großen Hauptbereich angezeigt werden soll, auf anderen hingegen in der kleineren Sidebar? Hier kommen Element Queries ins Spiel. Dr. Florence Maurice zeigt in einem Artikel ab Seite 64, wie man diese Technik einsetzen und dazu die JavaScript-Bibliothek EQCSS optimal nutzen kann.

Mit dem Open-Source-Programm Phan gibt es ein Tool, das sich PHP-Programme ansieht und mögliche Schwachstellen identifiziert. Da es eine statische Analyse durchführt, braucht es dabei nicht unbedingt einen Server für seine Tests. Besonders interessant: Es kann auch feststellen, ob die Codebasis bereits fit für PHP 7 ist, und liefert alle Stellen, die potenziell Ärger bereiten. Einer der beiden Hauptautoren von Phan ist übrigens Rasmus Lerdorf, also der Erfinder von PHP. Markus Schraudolph stellt das nützliche Werkzeug für PHP-Entwickler in einem ausführlichen Artikel ab Seite 116 vor.

Ihr Max Bold
chefredakteur@maxbold.de

Philip Ackermann

erläutert die sprachabhängige Formatierung von Datums- und Zeitangaben via API (S. 58)

Dr. Markus Stäuble

präsentiert die Versionsverwaltung Git im freien Online-Repository GitHub (S. 100)

Ender Özgür

zeigt die Must-have-Skills erfolgreicher Software-Entwickler und IT-Profis (S. 130)

INHALT

Update

Feature

HTML, CSS & JavaScript

Mobile Development

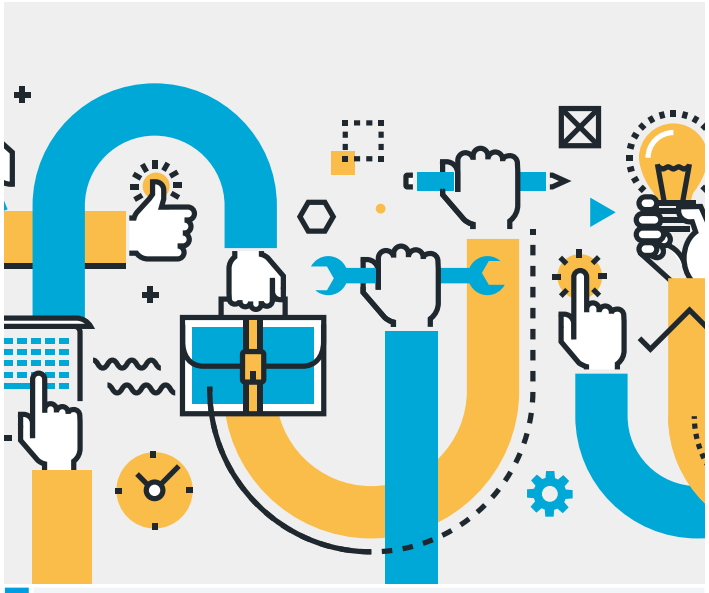


Foto: Matthias Vietmeier

Der Nutzen eines komponentenbasierten UI entsteht durch die Definition einer Designsprache 22

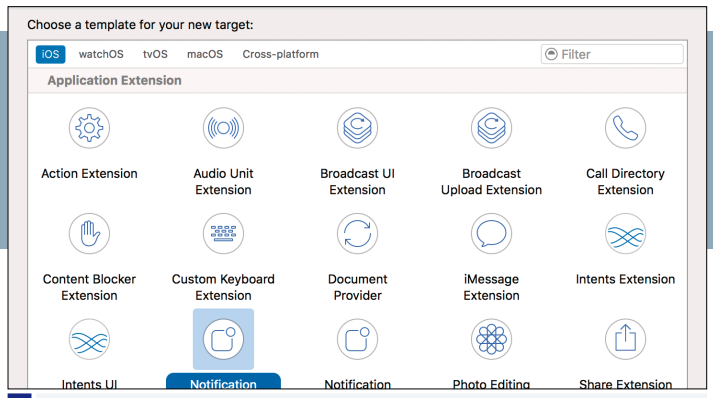
Foto: Shutterstock / Bleakstar

In JavaScript ist es dank dessen Dynamik deutlich einfacher als in anderen Sprachen, AOP zu implementieren 50

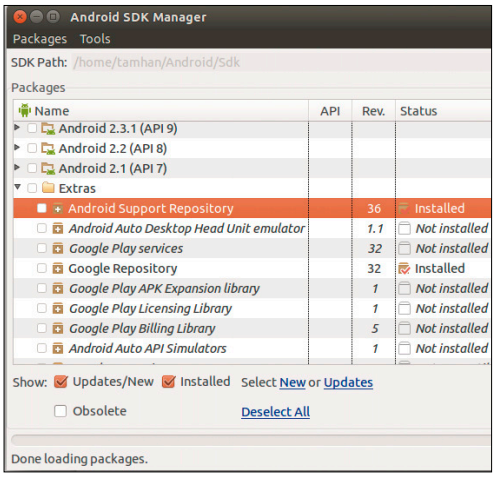
Experten in dieser Ausgabe

Philippe Bénard erläutert den Umgang mit Canvas- und SVG-Grafiken in HTML5 40

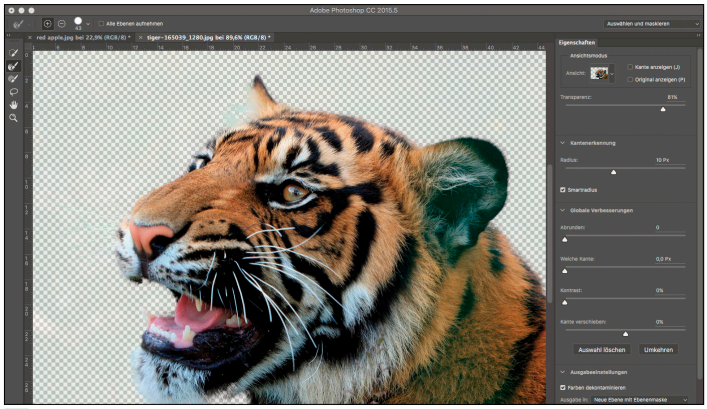
Ekkehard Gentz zeigt, wie man mit Qt Quick Controls 2 gebaute Apps schnell in die Stores bringt 84



Mit den Extensions Notification Content und Notification Service lassen sich die Inhalte von Notifications weiter optimieren **70**



Fehlersuche in Android-Apps: Das Android Support Repository ist für Instrumented Tests zwingend erforderlich **90**



Adobe Photoshop CC 2015: Auswählen und maskieren ist eine erweiterte Möglichkeit, beispielsweise Haare freizustellen **120**

NEWS & TRENDS

AKTUELLE NEWS FÜR ENTWICKLER

SQLite 3.14

Mit verbessertem CSV-Support

SQLite ist eine Programmbibliothek, die ein relationales Datenbanksystem enthält. Das System ist vor allem für den Einsatz in eingebetteten Datenbanksystemen entworfen worden.

Die gesamte Datenbank befindet sich in einer einzigen Datei. Eine Client-Server-Architektur ist deshalb nicht vorhanden. Ihr Konzept der virtuellen Tabellen erlaubt es, auf diverse Datenquellen mit SQL-Befehlen zuzugreifen. SQLite behandelt diese Datenquellen wie normale Tabellen.

Wer die gerade erschienene Version 3.14 von SQLite nutzt und die Standardvariante zusätzlich um die Datei *ext/misc/csv.c* erweitert, kann jetzt auch kommaseparierte Dateien (CSV-Dateien) wie SQL-Tabellen in seine Programme einbinden und mit SQLite lesen und schreiben.

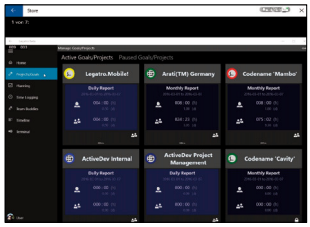
www.sqlite.com

In der neuen Version 3.14 kann SQLite CSV-Daten als virtuelle Tabellen verwalten

Legatro

Eine App für alle, die Stunden abrechnen

Wer für jedes seiner Projekte permanent die aufgewendeten Zeiten notieren muss und das noch manuell erledigt, sehnt sich nach einem geeigneten Werkzeug, das ihm dabei hilft. Mit der App Legatro kann man eigene Ideen, Aufgaben und Projekte auf komfortable Weise verwalten. Sie funktioniert mit jedem Tablet, Smartphone oder Desktop unter Windows 10 – auch Continuum für Windows Mobile wird unterstützt. Die



Für kleine Teams gibt es die App Legatro kostenlos

App hilft beim Aufzeichnen der Zeiten, liefert Berichte und bietet Export-Funktionen nach PDF und Excel. In der jüngsten Version ist auch eine Anbindung an eine Office 365 Business Subscription möglich.

legatro.net

Projektmanagement-Software In-Step Blue

Neue Version verfügbar

Die Berliner Microtool GmbH hat die Version 6.0.3 ihrer Projektmanagement-Software In-Step Blue veröffentlicht. Mit In-Step Blue werden Organisationen unterstützt, die in ihren Projekten auf Basis von Prozessen und Methoden arbeiten und Mitarbeiter mit ihren Rollen und Aufgaben in einer gemeinsamen Infrastruktur integrieren.

Neuer Webclient zur Bearbeitung von Daten und zum Verteilen von Dokumenten

Neben der Projektplanung und Projektsteuerung bietet die Software auch die Integration verschiedener Disziplinen wie Anforderungs- und Änderungsmanagement, Risiko- und Testmanagement, Dokumenten- und Multiprojektmanagement.

In-Step Blue 6.0.3 stellt einen neuen Webclient zur Verfügung, mit dem Anwender Dateien bearbeiten, Dokumente verteilen, Ereignisse auslösen sowie Sichten filtern und sortieren können.

Fortan können Kunden Referenzen in Formularen inline editieren sowie in Produktabhängigkeitshierarchien und Formularen filtern und verschieben. Die Anbindungen an Lotus Notes, Confluence und Active Directory wurden erweitert und die Unterstützung des Testmanagements kennt jetzt auch Testschritte.

PDF-Dateien lassen sich über eine neue Zustandsaktion mit Wasserzeichen versehen, und die Anzeige von Befehlen ist im Kontextmenü anpassbar. Auch die Geschwindigkeit der Systemkonvertierung im Zuge von Migrationen wurde verbessert.

www.microtool.de

Pirobase Imperia

CMS und PIM kombiniert

Die Pirobase Imperia GmbH hat auf der Dmexco ihre Pirobase Suite vorgestellt. Das Softwarepaket verzahnt ein Content-Management-System und ein Product-Information-Management-System miteinander.

»Durch die neue Pirobase Suite lassen sich valide Produktdaten aus dem PIM mit Marketinginhalten aus dem CMS verknüpfen, um sie in diverse Kanäle wie Websites oder Online-Kataloge, aber auch Händlernetzwerke oder Distributoren personalisiert auszuleiten«, erklärt Matthias Kant, Geschäftsführer der Pirobase Imperia GmbH, und ergänzt zudem: »Als Ergebnis arbeiten Redakteure stets mit aktuellsten Informationen für eine zielgerichtete Kommunikation. Dabei ermöglicht das intuitive Design selbst Redakteuren ohne tiefe IT-Kenntnisse eine einfa-

Matthias Kant, Geschäftsführer der Pirobase Imperia GmbH

Zahl des Monats

Zwischen Januar und Juni 2016 wurden weltweit **667 Millionen** Smartphones verkauft. Nach einem Plus von **7 Prozent** im ersten Halbjahr 2015 stieg die Zahl der verkauften Smartphones im ersten Halbjahr 2016 erneut um **6,2 Prozent.**

Quelle: GfK

che Bedienung.« Auch neu bei Pirobase Imperia sind Add-ons für Pirobase CMS: Social-Media-Integration, Analytics- sowie Umfrage-Tools für Erfolgsmessungen von Kampagnen.
www.pirobase-imperia.de

Combit

List & Label 22 kommt im Herbst

Zu den Top-Features der kommenden Version gehören laut Hersteller ausklappbare Bereiche, Top-N-Berichte und mehrere Ergebniszellen in Kreuztabellen. Außerdem sind für die neue Hauptversion zusätzliche Diagrammarten, Performanceverbesserungen und komfortable Funktionen für Webentwickler vorgesehen.

Insbesondere die Kreuztabellen im Berichtsdesigner erhalten laut Hersteller Combit eine große Palette an Neuerungen und Verbesserungen. Entwickler können den Berichtsdesigner mit der eigenen Applikation ohne weitere Lizenzgebühren verteilen. »Ausklappbare Bereiche, mehrere Ergebniszellen und Top-N-Berichte halten hier unter anderem Einzug«, so Jochen Bartlau, Entwicklungsleiter List & Label und Geschäftsführer bei Combit. »Mehr Abwechslung in Berichten gibt es durch die neuen Diagrammarten Radar und Treemap«, kündigt er weiterhin an. Zusätzlich erhalten Webanwendungen im Web Designer vielfältige neue Möglichkeiten. So können beispielsweise mit Projektbausteinen immer wiederkehrende Berichtselemente wie Ad- ▶

Bitkom

Gaming-Trends 2016

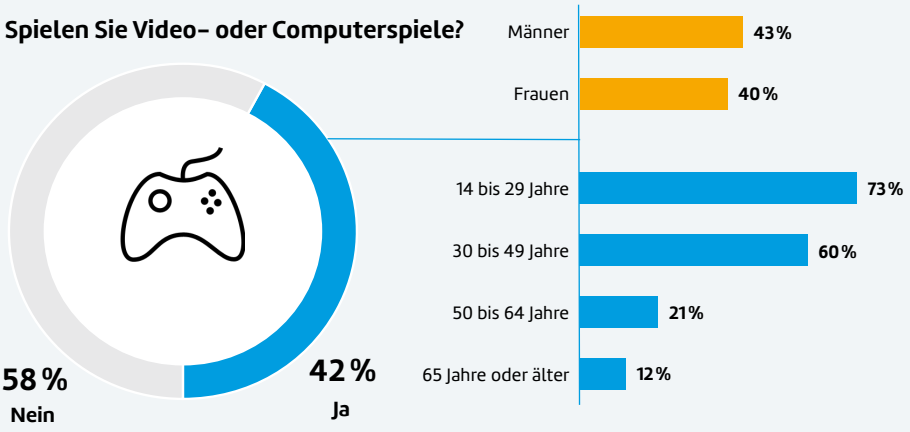
Zwei von fünf Bundesbürgern (42 Prozent) ab 14 Jahren beschäftigen sich regelmäßig mit Video- oder Computerspielen – das entspricht rund 30 Millionen Personen. Zu diesem Ergebnis kam eine repräsentative Befragung, die der Digitalverband Bitkom in Auftrag gegeben hat.

Die meisten Gamer finden sich dabei unter den Jüngeren: 71 Prozent der 14- bis 29-Jährigen spielen regelmäßig, unter den 30- bis 49-Jährigen sind es 60 Prozent. Von den 50- bis 64-Jährigen spielt jeder Fünfte (21 Prozent) und in der Generation 65 plus immerhin jeder Achte (12 Prozent).

Auch das Interesse der Nichtspieler an Gaming nimmt deutlich zu. Konnte sich vor zwei Jahren etwa jeder Fünfte (22 Prozent) vorstellen, künftig Computer- oder Videospiele zu spielen, ist es heute bereits jeder Dritte (32 Prozent). Unter den 14- bis 29-Jährigen, die bislang keine Computer- oder Videospiele spielen, sagen sogar mehr als die Hälfte (61 Prozent), dass sie sich vorstellen können, das künftig zu tun. Von den Befragten über 65 kann sich immerhin jeder Fünfte (21 Prozent) vorstellen, künftig selbst zu spielen.

Befragt nach den Gründen, weshalb sie nicht spielen, sagen 6 von 10 Befragten (60 Prozent), dass ihnen die Zeit dazu fehlt. 57 Prozent geben an, dass ihnen das Spielen keinen Spaß macht. Und 7 Prozent sind der Ansicht, dass Video- und Computerspiele nur etwas für Kinder sind. Im Vorjahr teilten diese Meinung noch 12 Prozent. »Gaming hat sich als Freizeitbeschäftigung für Erwachsene etabliert und wird auch als solche wahrgenommen«, so Martin Börner., Präsidiumsmitglied des Digitalverbands Bitkom.

Mobilgeräte sind als Gaming-Plattform besonders beliebt. 85 Prozent der Gamer spielen auf dem Smartphone. Auch Tablets setzen sich zunehmend als Spiele-Plattform durch: Während 2013 nur etwa jeder Zehnte auf dem Tablet spielte, ist es heute bereits jeder Zweite (52 Prozent). 41 Prozent der Gamer spielen auf einer mobilen Spielekonsole. Doch auch stationäre Geräte sind beliebt: Zwei Drittel (67 Prozent) der Gamer spielen am stationären PC und mehr als die Hälfte (59 Prozent) auf einer stationären Spielkonsole.
www.bitkom.org



Basis: Bevölkerung ab 14 Jahren (n = 1247)

Video- und Computerspiele haben sich als Freizeitbeschäftigung über alle Altersgruppen hinweg fest etabliert

web & mobile developer 11/2016

Quelle: Bitkom Research

Public-Cloud-Dienste

Turbosegmente der deutschen Internetwirtschaft

Insgesamt ist hier bis 2019 ein jährliches Umsatzwachstum von durchschnittlich 21,5 Prozent zu erwarten, wie eco – Verband der Internetwirtschaft e. V. und Arthur D. Little in ihrer gemeinsamen aktuellen Studie »Die deutsche Internetwirtschaft 2015 – 2019« zeigen. Vor allem die Public-Cloud-Dienste haben daran großen Anteil.

Die Public-Cloud-Dienste dürfen sich in diesem wie auch in den kommenden Jahren über ein immenses Wachstum freuen. Laut der Studie von eco und Arthur D. Little werden sie ihren Umsatz von 1,6 Milliarden Euro im Jahr 2015 auf 4,4 Milliarden Euro im Jahr 2019 steigern. »Alle Public-Cloud-Segmente werden deutlich an Bedeutung gewinnen«, sagt Andreas Weiss, Direktor EuroCloud Deutschland_eco e. V., »und zwar sowohl im B2B- als auch im B2C-Bereich.«

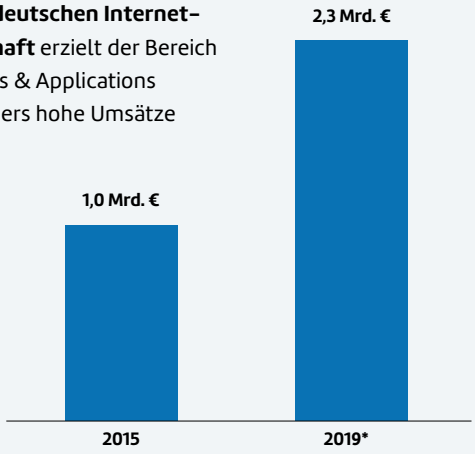
Unter den drei Public-Cloud-Segmenten erreicht Software as a Service (SaaS) seit jeher den Spitzenplatz – und wird ihn auch bis 2019 behaupten. Unter diesem Service versteht man die Möglichkeit, Anwendungen nach Bedarf über das Internet zu mieten. Im Jahr 2012 betrug der Umsatz noch 0,5 Milliarden Euro, im Jahr 2015 waren es bereits 1,0 Milliarden Euro, 2019 werden es 2,3 Milliarden Euro sein. In dem Zeitraum, den die Studie von eco und Arthur D. Little abdeckt, also von 2015 bis 2019, wächst dieses Segment damit um 23 Prozent jährlich.

Die hohe Wachstumsrate verdanken die SaaS-Anbieter dem Trend, dass immer mehr Mitarbeiter in den Unternehmen ebenso wie Privatnutzer stets und überall auf Anwendungen und Daten zugreifen können möchten. Mehrwerte von SaaS gegenüber herkömmlichen Lösungen – etwa Kostenvorteile und höhere Flexibilität – lassen sich zudem gut vermarkten.

30 bis 40 Prozent der Public-SaaS-Umsätze werden im Bereich B2C generiert. In den deutschen Unternehmen liegt die SaaS-Durchdringung derzeit bei rund 20 Prozent. »Sie wird in allen Industriezweigen in den kommenden Jahren deutlich zunehmen«, prognostiziert Andreas Weiss.

www.eco.de

In der deutschen Internetwirtschaft erzielt der Bereich Services & Applications besonders hohe Umsätze



* Prognose

web & mobile developer 11/2016
Quelle: eco / Arthur D. Little: Die deutsche Internetwirtschaft 2013 – 2019

ressdaten oder Firmenlogos zentral abgelegt werden.

Ein automatisch erstelltes Inhalts- und Indexverzeichnis erleichtert die Navigation im Bericht. Das vereinfacht die Berichtserstellung erheblich. Die Gestaltung interaktiver Berichte im Web Designer wird durch Drilldown ergänzt. Dargestellt werden sie anschließend im HTML5 Viewer. Das Reporting-Tool lässt Anwender Berichte mit visuellen Einblendungen sowie Übergängen animieren. So werden datengestützte Erkenntnisse bei der Präsentation noch besser verdeutlicht, und es wird kein separates Präsentationsprogramm benötigt.

List & Label 22: Die Software für CRM, Kontaktmanagement und Reporting für Entwickler

Software-Entwickler können ihre Anwendungen um leistungsstarke Reporting-Funktionen erweitern und haben Zugang zu jeder Art von Datenaufbereitung und Datenaustausch, in klassischen Anwendungen ebenso wie im Webreporting und in der Cloud.

www.combit.net

Raspberry Pi

Minicomputer im Unternehmen

Ursprünglich einmal als erschwinglicher Lerncomputer gestartet, hat sich der nur scheckkartengroße Raspberry Pi zu einer leistungsfähigen Entwicklungsplattform gemausert. Da das Gerät auch mit diversen Sensoren und elektronischen Bauteilen kommunizieren kann, eignet es sich besonders gut für Unternehmensanwendungen, die sonst nur mit deutlichen teureren Komponenten zu realisieren wären. Die kleine Platine muss sich im Zusammenspiel mit dem richtigen Display und einer geeigneten Grafikbibliothek nicht mehr hinter anderen Plattformen verstecken.

Eine mögliche Lösung dafür ist, passend zur Linux-Basis vieler Raspberry-Pi-Systeme, GTK+ 3. Damit lassen sich nicht nur komplexe Oberflächen gestalten, sondern auch komplexe Touch-Gesten einbinden.

Der IT-Lösungsanbieter Giegerich & Partner hat sich unter anderem auf die Anwendungsentwicklung mit Raspberry Pi spezialisiert. Zu den aktuellen Anwendungsbeispielen gehören die Steuerungssoftware sowie die Bedienoberfläche eines Fertigungsgeräts für Keramikelemente in der Zahnmedizin. Im Vordergrund der Entwicklung stand dabei vor allem die einfache und übersichtliche Bedienung über ein 7 Zoll breites Bedienpanel.

Giegerich & Partner sieht in der Entwicklung mit Raspberry Pi zahlreiche Vorteile im Vergleich zu konventionellen Systemen. Demnach erzielen entsprechende Lösungen eine deutlich bessere Performance. »Das liegt daran, dass die Entwicklung in der Regel für eine spezifische Auflösung erfolgt. Dadurch lassen sich viele dynamische Layouts für GUI-Elemente vermeiden – wie beim Beispiel des Fertigungsgeräts oder anderen Steuerungsdisplays von Geräten«, so Hans-Joachim Giegerich, Geschäftsführer von Giegerich & Partner, zu den Besonderheiten bei der Entwicklung. »Im Hinblick auf die Hardwarekosten ist ein Raspberry Pi mit rund 40 Euro unschlagbar«.

www.giepa.de

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

Red Hat

Virtualization 4 vorgestellt

Red Hat präsentiert mit Virtualization 4 die neue Version seiner KVM-Virtualisierungsplattform (Kernel-based Virtual Machine). Red Hat Virtualization 4 bietet eine offene, hochperformante, sichere und zentralisierte Management-Plattform für Linux- und Windows-basierte Workloads und stellt damit eine Alternative im Hinblick auf Wirtschaftlichkeit und Komplexität zu proprietären Virtualisierungslösungen dar. Sie beinhaltet einen leistungsstarken Hypervisor, ein fortschrittliches System-Dashboard und ein zentralisiertes Netzwerk-Management für steigende Workload-Anforderungen. Aufbauend auf Red Hat Enterprise Linux ist Red Hat Virtualization 4 für die einfache Integration in bestehende IT-Umgebungen konzipiert. Zudem unterstützt die Virtualisierungslösung auch die Nutzung neuer Technologien, einschließlich containerisierter und Cloud-nativer Applikationen.

Red Hat hat eine neue Version seiner KVM-Virtualisierungsplattform vorgestellt

Virtualisierung bleibt ein zentrales Element der Rechenzentrums-Infrastruktur, und die Anwenderanforderungen rund um die Technologie steigen rapide. Die neue Lösung von Red Hat richtet sich zum einen an Unternehmen, die eine Virtualisierungsstrategie einschlagen und dabei eine umfassende, agile Plattform suchen, die Effizienz und offene Standards für Interoperabilität bietet. Zum anderen ist die Lösung für Unternehmen konzipiert, die bereits Virtualisierungstechnologien nutzen, aber zunehmend aufgrund deren Kosten, Performance-Einschränkungen oder Inkompatibilität beunruhigt sind.
Red Hat Virtualization 4 adressiert diese Szenarien mit einer Plattform, die auf offenen Standards basiert und eine leistungsstarke, flexible Lösung für neue Implementierungen und die Migration vorhandener Virtualisierungsumgebungen bereitstellt.

Red Hat Virtualization 4 beinhaltet sowohl einen hochperformanten Hypervisor (Red Hat Virtualization Host) als auch einen webbasierten Virtualization Resource Manager (Red Hat Virtualization Manager) für die Verwaltung einer Virtualisierungs-Infrastruktur.

www.redhat.com/de

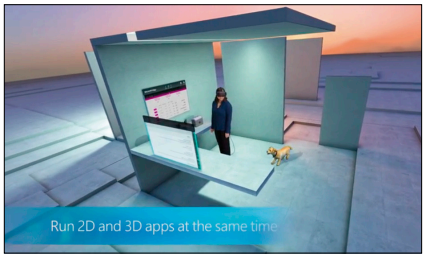
Intel und Microsoft

Windows Holographic Shell auf PCs

Microsoft und Intel haben auf dem Intel Developer Forum Spezifikationen für ihre Mixed-Reality-Vorhaben angekündigt. Die Holographic Shell wird Bestandteil von Windows und soll Oculus Rift und HTC Vive ansteuern können. Intel bringt einen Wettbewerber für Microsofts HoloLens.

Windows-Chef Terry Myerson und Intel CEO Brian Krzanich haben auf dem Intel Developer Forum (IDF) die Spezifikationen der beiden Unternehmen für Mixed Reality angekündigt. Sie demonstrierten die Windows Holographic Shell zum ersten Mal auf einem VR-Device und zeigten die Skalierbarkeit von Windows Holographic in Verbindung mit kostengünstigen PCs. Die auch im Produkt Microsoft HoloLens verwendete Technologie hält somit Einzug in Windows 10 – angeblich bereits im kommenden Jahr. Über ein an den Rechner angeschlossenes Headset lassen sich dann Windows-Systemfunktionen und Apps über die Brille nutzen.

Die Holographic Shell soll Oculus Rift und HTC Vive unterstützen. Außerdem kündigte Intel ein eigenes Headset namens Alloy an, das genau wie HoloLens über eine integrierte CPU/GPU und einen Akku verfügt, also unabhängig von einem PC arbeiten kann.



Holographic Shell wird Bestandteil von Windows

www.microsoft.com

Kaspersky-Studie

IT-Sicherheitsfachkräfte dringend gesucht

Großunternehmen, die bei der Suche nach gut ausgebildeten IT-Sicherheitsfachkräften scheitern, zahlen am Ende eine bis zu dreimal höhere Summe für die Bewältigung eines Cybersicherheitsvorfalls. Das geht aus einem Bericht von Kaspersky Lab auf Basis der weltweiten Studie »IT Security Risks Special Report Series 2016« hervor.

Entsprechend der Studie steigt bei einem signifikanten Teil der Unternehmen der Bedarf an Sicherheitsexperten. Vom Fachkräftemangel betroffene Firmen sind auch bereit, höhere Gehälter für immer weniger verfügbares Personal am Markt zu zahlen.

Eine komplexere IT-Infrastruktur, wachsende Compliance-Anforderungen und der steigende Bedarf an Schutzmaßnahmen für das Betriebsvermögen führen in vielen Unternehmen dazu, die interne Security Intelligence zu verbessern.

In jedem dritten Unternehmen (33 Prozent) gehören Verbesserungen spezieller Sicherheitsfachkenntnisse zu den drei wichtigsten Gründen für weitere Investitionen in die IT-Sicherheit. Allerdings stehen immer weniger Spezialisten zur Verfügung, die zudem mit ständig wachsenden Anforderungen konfrontiert werden. Kaspersky Lab beschäftigt selbst viele Hundert IT-Sicherheitsexperten. Nach Angaben der Personalabteilung von Kaspersky Lab erfüllt derzeit nur etwa jeder vierzigste Bewerber die strengen Kriterien

Holger Suhl, General Manager DACH von Kaspersky Lab

der ausgeschriebenen Expertenpositionen.

Auch die Anforderungen an Sicherheitsexperten wachsen. Neben einem tiefgehenden technischen Know-how werden laut Kaspersky Lab auch Management-Eigenschaften wie Kommunikationsfähigkeit und strategisches Denken benötigt.

Für eine erfolgreiche Ausbildung im Bereich IT-Sicherheit sind neben einer gewissen Leidenschaft für die IT auch eine selbstständige Fortbildungsbereitschaft sowie eine Anpassungsfähigkeit an sich ständig verändernde Bedrohungsszenarien von Bedeutung.

68,5 Prozent aller Unternehmen erwarten, dass die Anzahl der Mitarbeiter, die sich ausschließlich IT-Sicherheitsfragen widmen, steigen wird.

Gesucht werden insbesondere akademisch ausgebildete Experten, und hier ist die IT-Sicherheitsbranche auch selbst gefragt. Eine Möglichkeit wäre, Universitäten und Hochschulen mit entsprechender Sicherheitsexpertise zu unterstützen. Langfristig vielversprechend ist auch, dass die Security Intelligence – also die in einem Sicherheitsunternehmen vorhandene Expertise – in Form von Daten-Feeds, Cybersicherheitstrainings und sonstigen Services mit Unternehmenskunden geteilt wird.

Eine Kombination aus Security-Lösungen und Intelligence kann den IT-Sicherheits- ▶

Forsa-Studie 2016

Smartphone-Nutzung am Steuer

Rund zwei Drittel (64 Prozent) aller Autofahrer nutzen ihr Smartphone beziehungsweise Mobiltelefon zumindest hin und wieder am Steuer. Noch mehr (71 Prozent) sehen häufig oder sogar sehr häufig andere Verkehrsteilnehmer, die dies tun. Diese Zahlen sprechen für eine Normalität risikoreichen Verkehrsverhaltens, das von der Mehrheit der Autofahrer selbstkritisch zugegeben wird: 20 Prozent fühlen sich bei der eigenen Smartphone-Nutzung »sehr stark«, 40 Prozent »stark« abgelenkt. Dies besagt eine gerade erschienene repräsentative Umfrage des Meinungsforschungsinstituts Forsa im Auftrag des Automobil-Club Verkehr (ACV) und der Deutschen Verkehrswacht (DVW).

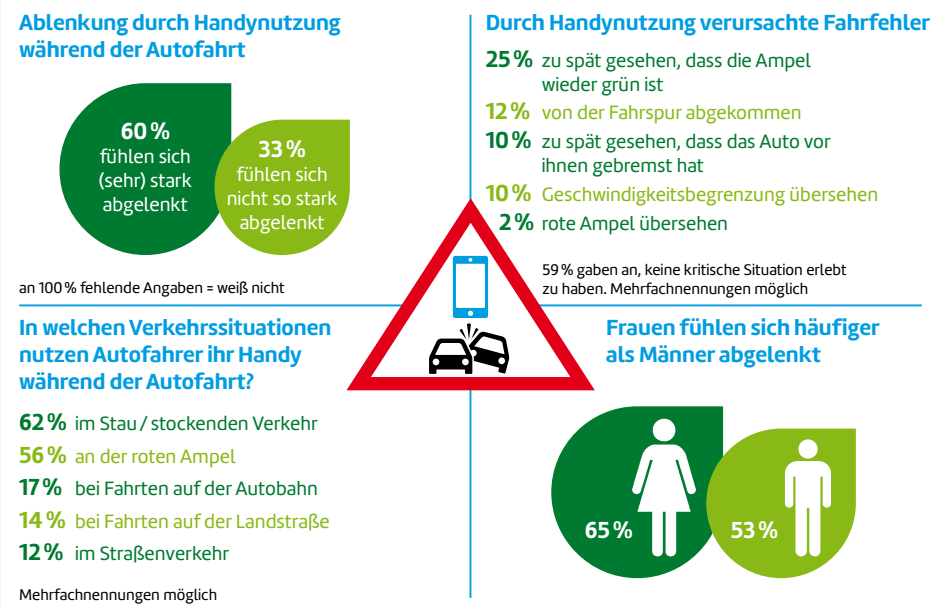
Die Umfrage beschäftigte sich insbesondere damit, wer das Smartphone/Mobiltelefon benutzt und in welchen Verkehrssituationen. Gefragt wurde ferner nach Situationen, in denen das Mobiltelefon auf keinen Fall genutzt würde. Auch interessierte die Auftraggeber, ob sich die Nutzer von ihrer eigentlichen Tätigkeit – dem Autofahren – abgelenkt fühlten und aufgrund der Nutzung kritische Verkehrssituationen entstanden sind oder Fahrfehler begangen wurden.

Die jungen Autofahrer bis 24 Jahre, die von allen Verkehrsunfallstatistiken als Hochrisikogruppe ausgewiesen werden, gehören zu den fleißigsten Nutzern des Mobiltelefons während des Autofahrens. Sie befinden sich dabei laut der Forsa-Umfrage in Gesellschaft mit den jüngeren Autofahrern bis unter 45 Jahren. Bis zu diesem Alter nutzen 81 Prozent das Smartphone zumindest hin und wieder während der Autofahrt – zum Beispiel zur Navigation, zum Nachsehen und Lesen, ob jemand angerufen oder eine SMS respektive WhatsApp geschrieben hat, zur Annahme von Anrufen (ohne Freisprecheinrichtung). Darüber hinaus wird das Telefon aber auch »proaktiv« genutzt, um beispielsweise selbst Anrufe zu tätigen (ohne Freisprecheinrichtung) oder eine SMS oder WhatsApp zu tippen. Die bis 29-Jährigen nutzen es auch sehr gern zur Musikauswahl (31 Prozent).

Erst Autofahrern ab 60 Jahren scheint das Mobiltelefon am Steuer entbehrlich zu sein. Hier sind es nur noch 38 Prozent, die angeben, es während des Autofahrens zumindest hin und wieder zu nutzen, dabei meist (19 Prozent) zur Navigation oder um nachzusehen, ob jemand geschrieben oder angerufen hat (15 Prozent). Auf eine proaktive Nutzung verzichten die meisten autofahrenden Senioren.

www.acv.de

Handy am Steuer – Ablenkung im Straßenverkehr



Die Umfrage untersucht, wer das Smartphone benutzt und in welchen Verkehrssituationen

Hosting für Gamer

Performance und Bandbreite sind am wichtigsten

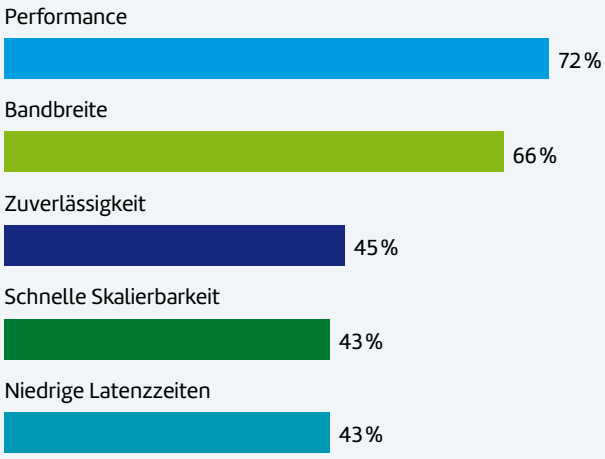
Eine hohe Performance und eine große Bandbreite stellen die wichtigsten Kriterien bei der Auswahl eines Hosting-Dienstleisters in der Gaming-Branche dar. Dieses Ergebnis hat eine aktuelle Umfrage von LeaseWeb Deutschland unter 238 Spiele-Experten zur Gamescom zutage gefördert. Demnach stellt die Performance für 72 Prozent und die Bandbreite für 66 Prozent der Befragten ein K.-o.-Kriterium beim Hosting dar.

Als weitere wesentliche Aspekte bei der Wahl eines Hosting-Partners hat die Umfrage ergeben: hohe Zuverlässigkeit (45 Prozent), schnelle Skalierbarkeit und niedrige Latenzzeiten (43 Prozent) sowie maximale Flexibilität, die Möglichkeit einer kundenspezifischen Konfiguration und Erfahrung in der Gaming-Branche (41 Prozent). Danach folgen eine hohe Verfügbarkeit und ein gutes Preis-Leistungs-Verhältnis (38 Prozent) mit einem fairen Pricing entlang der Skalierung (31 Prozent) sowie ein 24/7-Rund-um-die-Uhr-Support und -Service (34 Prozent). Nicht ganz so wichtig sind der Gaming-Branche eine vollständig redundante Architektur (29 Prozent) und eine weltweite Infrastruktur mit Datenzentren und Zugangsknoten auf allen Kontinenten (21 Prozent).

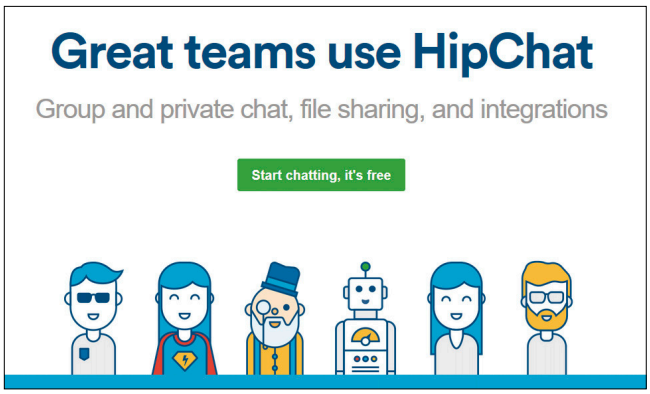
Auf jeden Fall muss die Hosting-Infrastruktur für Multiplayer-Echtzeit-Games geeignet sein, meinen 90 Prozent der Befragten. Für 61 Prozent stehen virtuelle Welten im Mittelpunkt. Weiterhin vorausgesetzt wird eine Eignung der Infrastruktur für E-Sport mit Millionen von IP-TV-Zuschauern (48 Prozent), Actionspiele (46 Prozent) und Simulationsspiele (45 Prozent). Laut Umfrage weniger relevant sind Adventure Games (32 Prozent), Fantasy-Spiele (31 Prozent), Casual Games (26 Prozent), Free-to-Play-Games (24 Prozent), Jump-and-Run-Spiele (21 Prozent) sowie naturgemäß Strategiespiele (14 Prozent).

Bei der Infrastruktur bevorzugen die meisten Umfrageteilnehmer On-Demand-Bare-Metal-Server (54 Prozent) und virtuelle Server (50 Prozent) sowie dedizierte Server (41 Prozent). Cloud-Lösungen akzeptieren nur 29 Prozent der Teilnehmer.

www.leaseweb.de



LeaseWeb hat 238 Spiele-Experten nach ihren Anforderungen in Sachen Hosting befragt



Die Software erlaubt spontane Gruppen-Videokonferenzen sowie Screen-Sharing

teams in den Unternehmen dabei helfen, sich mehr auf die Entwicklung einer Cybersicherheitsstrategie und auf hochentwickelte Bedrohungen zu konzentrieren, statt einzelne IT-Sicherheitsvorfälle zu bearbeiten.

»Cybersicherheit bedeutet mehr als Sicherheitstechnologie«, sagt Holger Suhl, General Manager DACH von Kaspersky Lab. »Wir müssen unseren Kunden über Trainings die nötigen Fähigkeiten vermitteln, Cyberangriffe zu erkennen. Dazu werden detaillierte Kenntnisse über Angriffe auf andere Unternehmen benötigt, etwa in Form kundenspezifischer Berichte sowie über automatisierte Bedrohungsinformationen. Um die verschiedenen Herausforderungen bei der Gefahrenabwehr zu lösen, erfordert die Erkennung zielgerichteter Attacken sowie Vorhersage und Reaktion auf Cyberangriffe viel Flexibilität. Als Anbieter von Sicherheitslösungen werden wir unser IT-Sicherheitspersonal hinsichtlich Qualität und Quantität weltweit weiter ausbauen.«

www.kaspersky.com/de

Atlassian

Gruppen-Videochat für Teams

Das neue Produkt von Atlassian ist ein Aufsatz für HipChat Plus und erlaubt spontane Gruppen-Videokonferenzen sowie das Screen-Sharing.

Im Vergleich zu Consumer-Produkten wie Google Hangouts, Apple Facetime oder dem Facebook Messenger ist das Aufsetzen eines Team-Videochats für sichere Face-to-Face-Besprechungen immer noch recht aufwendig.

Dies will Atlassian mit seinem Produkt HipChat und der neu darauf aufsetzenden HipChat-Video-Plattform ändern. Seit ein paar Tagen steht HipChat-Plus-Anwendern die neue Videochat-Funktion zur Verfügung, die zudem das Screen-Sharing erlaubt – laut Atlassian sogar ohne zusätzliche Kosten.

Wie Atlassian betont, ist die neue Videoplattform genauso stabil wie die verlässlichsten Plattformen auf dem Markt, sie lässt sich aber mit nur einem Mausklick einrichten.

Es muss dafür keine Agenda angelegt und kein Zeitfenster reserviert werden. Sind alle Teammitglieder im HipChat versammelt, genügt es, die Videokonferenz zu starten. Niemand muss eine zusätzliche Software installieren oder den Chat-Raum wechseln.

Für Kunden von HipChat Plus kommt die neue Funktion mit dem Upgrade ihrer Desktop-Apps. Für HipChat-Server-Kunden soll die Erweiterung demnächst ausgerollt werden.

www.hipchat.com

DOMAINS | E-MAIL | **HOSTING** | SHOPS | SERVER

NEU! 1&1 MANAGED CLOUD HOSTING

Das beste aus zwei Welten!

Jederzeit skalierbare und flexible Server-Ressourcen kombiniert mit einem leistungsstarken Hostingpaket: das **neue 1&1 Managed Cloud Hosting ist da!** Ideal für Online-Projekte mit höchsten Ansprüchen an Verfügbarkeit, Sicherheit und Flexibilität.

- ✓ **Dedizierte Ressourcen**
- ✓ **20+ Stack Variationen**
- ✓ **Von 1&1 Experten gemanagt**
- ✓ **Flexibel skalierbar**
- ✓ **Bereitstellung < 1 Minute**

Cloud Vendor Benchmark 2016
Rising Star Germany
experton GROUP

intel XEON inside

Trusted Performance.
Intel® Xeon® processors.

ab 9,99 €/Monat*

Internet made in Germany

1 MONAT VERTRAGS-LAUFZEIT

1 ANRUF EXPERTEN-RAT

1 GUTES GEFÜHL AUSFALLSICHER

DE: 02602/9691
AT: 0800/100668

1&1

1und1.info

*1&1 Managed Cloud Hosting ab 9,99 €/Monat. Kündigung im ersten Monat jederzeit möglich. Keine Bereitstellungsgebühr. Preise inkl. MwSt. 1&1 Internet SE, Elgendorfer Straße 57, 56410 Montabaur

Kaspersky-Studie

Die Deutschen und ihre digitale Zukunft

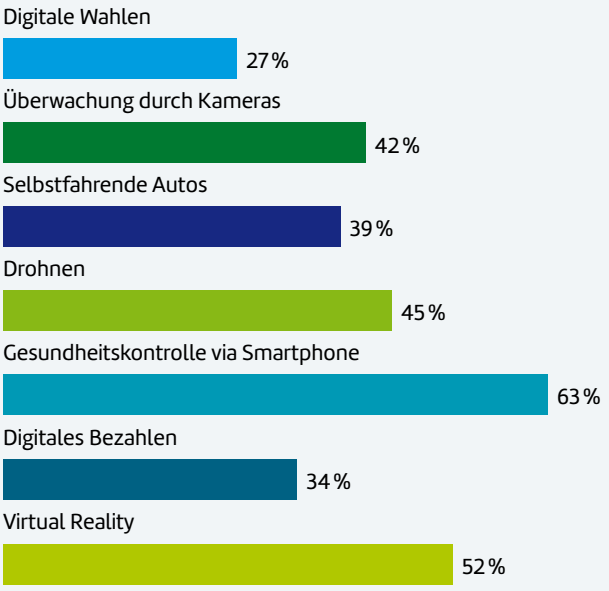
Vom smarten Kühlschrank über digitale Fitnesshelfer bis hin zu Virtual-Reality-Brillen, Connected Cars oder Drohnen – das ist die digitale Zukunft. Doch wie sehen die Deutschen ihren digitalen Alltag der Zukunft?

Laut einer Studie von Kaspersky Lab und Statista sieht sich eine Mehrheit der deutschen Befragten zukünftig noch nicht in der virtuellen Realität und bewertet das digitale Bezahlen nicht nur positiv. Außerdem hegt sie Skepsis gegenüber selbstfahrenden Autos und lehnt den Einsatz von Drohnen ab. Hingegen würde sie gerne digital wählen. Ein Teil kann sich sogar für Sex in virtuellen Welten begeistern. Die Themen Datenschutz und Cybersicherheit bereiten den Deutschen jedoch Sorge.

»Unsere Studie zeigt ein umfassendes Bild über die Einstellungen der deutschen Nutzer hinsichtlich ihres zunehmend digitaler werdenden Alltags«, sagt Holger Suhl, General Manager DACH bei Kaspersky Lab. Generell hält sich die Vorfreude bei den deutschen Nutzern auf die digitale Zukunft noch in Grenzen. So offenbarte die Kaspersky-Umfrage, dass 42 Prozent der deutschen Befragten ihr Gefühl diesbezüglich mit Unsicherheit, Unbehagen oder Angst beschreiben würden. Neugierig sind 40 Prozent. Lediglich sechs Prozent hegen große und 12 Prozent zurückhaltende Freude gegenüber dem zunehmend digitaler werdenden Alltag.

Die in Deutschland vorhandene Skepsis könnte auch mit den Themen Cybersicherheit und Datenschutz zusammenhängen. So ist über die Hälfte (56 Prozent) der befragten Deutschen der Meinung, ihr digitales Leben der Zukunft werde unsicherer; auch denken 69 Prozent, dass es sie gläsern mache. Demgegenüber behaupten 19 Prozent, ihr digitales Leben werde sicherer, und 18 Prozent, das digitale Leben schaffe Transparenz.

www.kaspersky.com/de



Von Skepsis bis zur Ablehnung reicht die Meinung der Deutschen zu manchen Ausprägungen der digitalen Zukunft

Quelle: Kaspersky

Microsoft

Windows 10 Enterprise im Abo für KMUs

Über das neue Modell E3 ist die Enterprise-Edition von Windows 10 ab dem 1. September 2016 für deutsche Unternehmenskunden erstmals im Rahmen eines monatlichen Abonnements via Cloud Solution Provider (CSP) verfügbar. Den Monatspreis gibt Microsoft mit 5,90 Euro pro Nutzer an. Voraussetzung für die Nutzung des Angebots ist eine gültige Version von Windows 10 Pro mit installiertem Anniversary Update auf dem betreffenden Gerät. Über den Login des Kunden via Azure Active Directory werden die bereits im System vorhandenen Enterprise-Funktionen dann einfach aktiviert.

Um Windows 10 Enterprise via CSP zu lizenzieren, müssen Kunden eine Lizenz für die Mindestlaufzeit von einem Jahr abonnieren. Die damit erworbene Lizenz umfasst fünf Geräte pro Anwender. Werden weitere Lizenzen benötigt, können diese dem Abonnement monatlich hinzugefügt oder wieder entfernt werden. Endet das Abonnement oder wird es ausgesetzt, wechselt das Gerät nach 90 Tagen zurück auf Windows 10 Pro. Zum Schutz vor Datenverlusten bei versehentlich ausgesetzten Abonnements bleiben wichtige Daten und Funktionen auch nach dem Ablaufen der Frist weiter im Hintergrund erhalten, können bis zur Erneuerung des Abonnements aber nicht mehr genutzt werden.

Gerade Organisationen aus den Bereichen Gesundheitswesen, Handel, Finanzdienstleistungen oder Produktion haben einen gestiegenen Bedarf an den umfangreichen Sicherheitsfunktionen, die Windows 10 Enterprise mit sich bringt. Zudem macht das Modell gerade die Anschaffung von Lizenzen noch einfacher und vor allem flexibler möglich. So haben CSP-Partner ab sofort die Möglichkeit, ihren Kunden das gesamte Spektrum von Microsofts Cloud-Lösungen aus einer Hand anzubieten: Von Windows 10 über die Enterprise Mobility Suite (EMS) bis hin zu Office 365 und Dynamics CRM Online.

Gleichzeitig profitieren Unternehmenskunden mit Windows 10 Enterprise E3 auch von den Vorteilen einer von erfahrenen Partnern verwalteten IT:

Für kleine und mittelständische Unternehmen gibt es Windows 10 Enterprise im Abonnement

Microsoft-Partner stellen die Abonnements bereit und helfen ihren Kunden darüber hinaus auch bei der Verwaltung der Hardware. Somit fallen durch das Partner-Modell beim Wechsel von Windows 10 Pro auf Windows 10 Enterprise keine weiteren Aufgaben für die eigene IT an. Vielmehr profitieren Firmenkunden auch von einer Vereinheitlichung im Bereich der Verwaltung – beispielsweise durch nur noch einen Vertrag, einen Nutzer-Account, einen Anlaufpunkt für Support-Anfragen und eine Rechnung.

www.microsoft.com

In-Memory Data Grid

Hazelcast jetzt im Azure Marketplace

Hazelcast stellt seine Open-Source-Version von Hazelcast 3.6.3 ab sofort als Image im Microsoft Azure Marketplace zur Verfügung. Die Software stellt eine In-Memory-Datenschicht bereit, die auf Bedarf elastisch skalierbar ist. Cloud-Entwickler können Hazelcast problemlos in ihre Anwendungen einbinden und damit die gleiche Skalierbarkeit und In-Memory-Leistung wie im Hochfrequenzhandel erzielen. Hazelcast im Microsoft Azure Marketplace zeichnet sich durch hohe Verfügbarkeit in der Cloud aus. Möglich wird dies durch übergreifende Partitionsgruppen in den Microsoft Azure-Regionen. So profitieren Kunden von Microsoft Azure von hochverfügbaren Anwendungen und nahtlosem Failover, da sich ein Cluster stets in einer oder in mehreren Azure-Regionen befindet. Falls eine Region verloren geht, ist der verbleibende Cluster immer noch groß genug zur Bewältigung der Workload. Dies ist insbesondere für Unternehmen von Bedeutung, die auf eine hohe Verfügbarkeit angewiesen sind.

Da Cloud-Entwickler und -Betreiber auf eine stärkere Nutzung von Open-Source-Lösungen drängen, hat Hazelcast das Cloud Discovery SPI eingeführt, das cloudbasierten Knoten oder Knoten vor Ort ermöglicht, sich innerhalb desselben Clusters untereinander automatisch zu erkennen. Der Konfigurations- und Verwaltungsaufwand wird dadurch minimiert.

Greg Luck ist CEO von Hazelcast

Neben der Unterstützung von Microsoft Azure bietet Hazelcast zudem auch Integrationsoptionen für Cloud-Management und -Konfiguration an, beispielsweise für Apache jclouds, AWS, Microsoft Azure, Consul, Eureka, Jetty Web Sessions, Kubernetes, Mesos, Tomcat Web Session und Zookeeper. Darüber hinaus umfasst Hazelcast Container-Deployment-Optionen für Docker, Cloud Foundry und OpenShift. Mit dem Ziel einer maximalen Erweiterbarkeit für individuelle Cloud-Strategien hat die Open Source-Community von Hazelcast entsprechende Client-APIs für eine Vielzahl von Programmierumgebungen entwickelt.

»Viele unserer Kunden entscheiden sich bei der Migration ihrer Infrastruktur in die Cloud für Microsoft Azure«, so Greg Luck, CEO von Hazelcast. »Gemeinsam mit Microsoft haben wir das Cloud Discovery Plug-in für Microsoft Azure entwickelt, sodass Hazelcast-Cluster in Azure ausgeführt werden können. Ab sofort steht Hazelcast unseren Kunden auch im Azure Marketplace zur Verfügung. Hierbei handelt es sich um eine vollständig getestete Integration mit Unterstützung von Hazelcast, Hazelcast Enterprise with BYOL und dem Management Center. Die gesamte Funktionalität von Hazelcast wird unterstützt. Dazu zählt auch das Deployment von Anwenderprogrammen bei Cluster-Mitgliedern. Wir freuen uns über diese Entwicklung und sehen unserem Angebot für Community und Kunden auf Microsoft Azure mit großem Interesse entgegen.«

www.hazelcast.com

Neue Netgear-LTE-Modems

Unternehmen vernetzen

Unternehmen mit Vertriebsteams oder Technikern im Außendienst profitieren von den neuen mobilen Breitbandmodems, die ihnen primäre oder Failover-Internetkonnektivität bereitstellen. Geschäftskritisches Equipment und wichtige Services können so rund um die Uhr betrieben werden.

Die Basisversion des neuen Netgear-LTE-Modems heißt LB1110 und richtet sich an Privatanwender und kleine Unternehmen. Das LB1111 zielt auf professionelle Anwender und bietet Power over Ethernet (PoE). Das LB1111 liefert sowohl Netzwerkkonnektivität als auch Strom über ein einziges Ethernetkabel und ist ideal für Außenstellen oder größere Büros geeignet, die mit PoE-Infrastruktur ausgestattet sind. Beide LTE-Modems verfügen über einen Gigabit-Ethernet-Port und ver-

Netgear hat auf der IFA 2016 zwei neue LTE-Modems vorgestellt

sorgen als Netzwerk-Bridge WLAN-Router, Gateways, Switches, Rechner, Sicherheitskameras oder andere Ethernet-fähige Geräte mit einer 4G-LTE-Verbindung. Darüber hinaus können Router, Gateways oder Switches die mobile 4G-Breitbandverbindung im Netzwerk teilen, sobald sie an den Gigabit-Ethernet-Port des LTE-Modems angeschlossen wurden. Damit steht die Breitbandgeschwindigkeit auch den verkabelten und den WLAN-Geräten im Netzwerk zur Verfügung.

»LTE setzt sich weltweit immer mehr durch und ist der einfachste Weg, mobilen Teams in Minuten eine Internetverbindung zur Verfügung zu stellen oder eine Failover-Lösung zu schaffen, die Schluss macht mit Internetausfällen«, erklärt Andrew Green, Vice President of Mobile Network Products bei Netgear. »Wenn eine kabelgebundene Lösung nicht möglich oder nicht gewünscht ist, oder der Kunde älteres Equipment für ein Pop-up-Büro einsetzen möchte, dann dienen die neuen Netgear-LTE-Modems als Bridge zwischen den Ethernet-Geräten und dem Internet.«

Größere Büros oder Außenstellen, die mit PoE-Infrastruktur ausgestattet sind (zum Beispiel den Netgear ProSafe PoE/PoE+ Switches), können das PoE-fähige LB1111-LTE-Modem einsetzen, um LTE-Zugriff in der gesamten Niederlassung zur Verfügung zu stellen. Die PoE-Funktion macht Steckdosen oder das Verlegen neuer Leitungen überflüssig und erleichtert Installateuren das Einrichten von Videoüberwachung, Wireless-Access-Points und VoIP bei ihren Kunden.

www.netgear.de

Bain-Studie

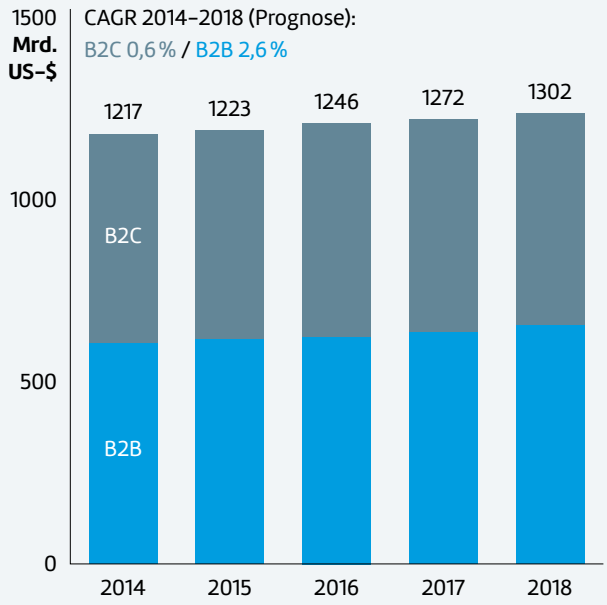
Wachstum durch B2B-Geschäft

In Westeuropa und Nordamerika stehen Telekommunikationskonzerne vor einem Strategiewechsel. Sie werden sich in Zukunft wieder mehr auf ihre Geschäftskunden konzentrieren und weniger auf das Privatkundensegment.

Der Grund: Das B2C-Geschäft wird bis 2018 nur noch um 0,6 Prozent zulegen, der B2B-Markt hingegen um 2,6 Prozent. Hauptwachstumstreiber im B2B-Markt sind mobile Datenlösungen und IT-Services mit einem Plus von 3,9 beziehungsweise 3,2 Prozent. Zu diesem Ergebnis kommt die aktuelle Studie »How to Capture the B2B Growth Opportunity in Telecom« der internationalen Managementberatung Bain & Company. Dr. Tobias Umbeck, Partner bei Bain & Company und Telekommunikationsexperte, betont: »Die Manager der Telekommunikationsanbieter müssen umdenken, die Wachstumsmöglichkeiten jenseits des traditionellen Konsumentenmarkts erkennen und ihre Unternehmen dahin steuern.«

Viele Telekommunikationskonzerne haben ihr B2B-Geschäft bisher vernachlässigt und sich zu wenig um die ertragreichen SOHO-Kunden gekümmert. Gerade diese Sparten bieten jedoch die größten Chancen, die Kundenabwanderung zu stoppen, mehr profitablen Umsatz zu erzielen und die Umstellung auf die moderne IP-Technologie zu forcieren. Dieser Strategiewechsel bedeutet für die meisten Telekommunikationskonzerne einen vollständigen Umbau ihres B2B-Geschäfts. »Wer zu lange mit der Überarbeitung seines Geschäftsmodells wartet, läuft Gefahr, die B2B-Revolution zu verschlafen«, warnt Bain-Partner Umbeck. »Nur wer seine Geschäftskunden kennt, wird auch in Zukunft noch zu den Marktführern gehören.«

www.bain.de



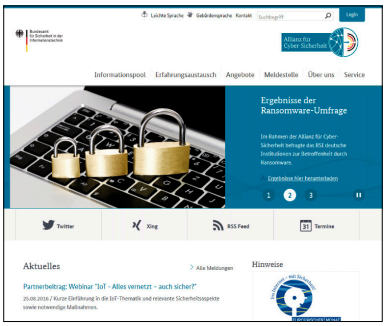
IT-Services und mobile Telekommunikations-Dienstleistungen werden das Umsatzwachstum des B2B-Geschäfts beschleunigen

 Quelle: Bain & Company

Ransomware

Die Bedrohung Nummer 1

Wer ist in welchem Ausmaß von Ransomware betroffen? Das wollte die Allianz für Cybersicherheit im Rahmen einer Umfrage überprüfen. Zwar gaben 68 Prozent Prozent der befragten Institutionen an, innerhalb der letzten sechs Monate nicht von Ransomware betroffen gewesen zu sein. Das übrige Drittel der Befragten war aber nicht nur betroffen, sondern 29 Prozent waren sogar das Ziel von mehr als einer Ransomware-Familie. Ebenso stellte sich heraus, dass Unternehmen aller Größenordnungen Opfer wurden. 54 Prozent der Unternehmen und Institutionen, die es bereits mit Erpresser-Software zu tun bekommen hatten, beschäftigen Mitarbeiter in einer Größenordnung von 1.000 bis 10.000, dicht gefolgt von der typischen Größe eines mittelständischen Unternehmens mit zwischen 250 und 999 Mitarbeitern. Unternehmen dieser Größe waren immerhin noch zu 44 Prozent von Ransomware-Angriffen betroffen.



In einer Umfrage wurden die Gefahren durch Ransomware eruiert

Die Ergebnisse der Umfrage decken sich hier mit denen anderer Umfragen und Studien des BSI. Deutlich angeführt wird die Statistik wenig überraschend von Locky, einer Variante, die Daten auf lokalen Laufwerken und Netzwerkfreigaben verschlüsselt und gerade in den USA und Deutschland besonders erfolgreich war. 93 Prozent der betroffenen Firmen/Institutionen bekamen es mit diesem Erpressungs-Trojaner zu tun, gefolgt von TeslaCrypt mit 74 Prozent, CryptoWall mit 24 Prozent und Sonstigen mit 27 Prozent. TeslaCrypt wurde erstmals im Februar 2015 veröffentlicht und richtet sich gegen Windows-Versionen inklusive Windows XP, Windows Vista, Windows 7 und Windows 8.

Der wichtigste Angriffsvektor, sofern er überhaupt bekannt ist, sind E-Mail-Anhänge. Sie waren bei der überwiegenden Mehrzahl der Betroffenen dafür verantwortlich, dass die Ransomware es ins System geschafft hat. Mit deutlichem Abstand folgen sogenannte Driven-by-Angriffe.

Derzeit gilt Ransomware weithin als raffinierte Angriffsmethode, gegen die kaum ein Kraut gewachsen ist. Nicht nur hat sie sich als erfolgreiches Geschäftsmodell für Hacker etabliert und die einzelnen Varianten es bereits zu einer gewissen Markenbekanntheit gebracht. Darüber hinaus sind professionelle Ransomware-Entwickler ausgesprochen findig und statten bekannte Familien mit immer neuen Features aus. Cerber verfügt nun beispielsweise über DDoS-Funktionen, und Microsoft berichtete von einer neuen Ransomware-Variante, die sich wie ein Computerwurm verbreiten kann. Und selbst Mac-Anwender sind längst nicht mehr auf der sicheren Seite. Die Zeiten, in denen Cyberkriminelle sich ausschließlich auf Windows-Betriebssysteme beschränkten sind seit KeRanger vorbei.

www.allianz-fuer-cybersicherheit.de

Jetzt kostenlos testen!

2 x gratis!

Das Fachmagazin für .NET-Entwickler

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

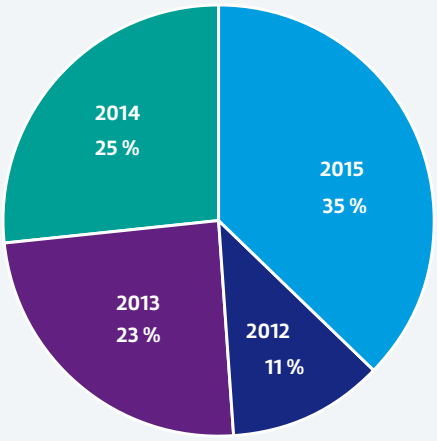
probeabo.dotnetpro.de/kostenlos-testen/

Shopping per Smartphone

Einkauf bevorzugt mobil

Das Smartphone wird künftig das wichtigste Einkaufsinstrument sein. Der Anteil der Kunden in Deutschland, die mindestens einmal monatlich einen Kauf über ihr Smartphone tätigen, ist in den vergangenen vier Jahren von 11 auf 35 Prozent gestiegen. Im selben Zeitraum ist der Anteil der Konsumenten, die noch nie mobil gekauft haben, von 70 auf 48 Prozent gesunken. Bis zum Jahr 2020 wird Mobile Shopping voraussichtlich für 75 Prozent der Käufer zumindest in ausgewählten Produktkategorien selbstverständlich zum Alltag gehören. Vorangetrieben wird diese Entwicklung vor allem von den Konsumenten zwischen 18 und 34 Jahren. Das sind Ergebnisse der Studie »Total Retail 2016 – Der Wettlauf um Relevanz« der Wirtschaftsprüfungs- und Beratungsgesellschaft PwC, für die 1000 deutsche Online-Shopper befragt worden sind. Die Analyse ist Teil einer weltweiten Umfrage mit mehr als 23.000 Teilnehmern in 25 Ländern.

»Im Mobile Shopping liegt für den deutschen Handel noch großes Potenzial«, sagt Gerd Bovensiepen, Leiter des Bereichs Handel und Konsumgüter bei PwC. »Nur die Unternehmen, die spätestens jetzt eine Mobile-Commerce-Strategie entwerfen



Die Vorliebe für den Einkauf über das Smartphone steigt deutlich

web & mobile developer 11/2016 Quelle: PwC

und diese beständig weiterentwickeln, werden auch in Zukunft erfolgreich sein.«

Ein wichtiger Indikator für die Entwicklung des Mobile Shopping ist der chinesische Markt. Die Befragung zeigt: Das Einkaufsverhalten der chinesischen Konsumenten ist dem der deutschen Verbraucher um etwa drei bis vier Jahre voraus. Ein deutlicher Unterschied zwischen dem deutschen und dem chinesischen Markt besteht in der Zahlweise. Während viele chinesische Verbraucher bevorzugt per Smartphone zahlen, sind deutsche Kunden in diesem Punkt wesentlich zurückhaltender: Nur 6 Prozent der deutschen Konsumenten wählen das Smartphone als bevorzugtes Zahlungsinstrument; sie zahlen lieber bar (80 Prozent), mit EC-Card (69 Prozent) oder auf Rechnung (50 Prozent). Der Grund für die Skepsis: Die deutschen Konsumenten befürchten, dass ihre Kontodaten nicht ausreichend geschützt sein könnten.

www.pwc.de

AVM

Neue Produkte für Smart Home

AVM hat auf der IFA einen intelligenten Heizkörperregler vorgestellt. FritzDECT 300 regelt die Temperatur im Haus und lässt sich per Webbrowser und App im Heimnetz und von unterwegs steuern.

Auch die smarte Steckdose FritzDECT 210 wurde präsentiert. Sie ist eine Outdoor-Variante der beliebten intelligenten Steckdose FritzDECT 200 für

Neue Smart-Home-Geräte von AVM.

das Schalten und Messen angeschlossener Geräte in Haus und Garten. Nützliche neue Funktionen hat die MyFritzApp 2 für Android erhalten. Als Fenster zur FritzBox ist sie jetzt zu Hause und unterwegs einsetzbar und noch einfacher zu bedienen. Anwender können sich über Anrufe, Sprachnachrichten oder Ereignisse im Heimnetz informieren lassen und Smart-Home-Geräte steuern. Mit einer FritzBox in Verbindung mit den soliden Fritz-Produkten lässt sich ganz leicht und ohne hohe Zusatzkosten ein Smart Home erstellen. Alle Fritz-Produke sind schnell ins Heimnetz eingebunden und lassen sich über PC, Notebook oder Smartphone intuitiv bedienen.

Durch das Betriebssystem FritzOS erhalten alle Fritz-Produkte regelmäßig Updates für neue Funktionen und mehr Sicherheit. Mit dem nächsten FritzOS wird das Fritz-Zuhause noch smarter. Die FritzFon-Modelle erhalten einen neuen Smart-Home-Startbildschirm und bei der intelligenten Steckdose FritzDECT 200 wird lokal am Gerät die Geräuscherkennung aktiviert. Mit HAN FUN, der geplanten Erweiterung des DECT-ULE-Standards, ermöglicht AVM die Integration von Smart-Home-Geräten anderer Hersteller in das Fritz-Heimnetz.

www.avm.de

Gewerkschaft

Beschäftigte durch Digitalisierung mehr belastet

Der zunehmende Einsatz digitaler Technik bringt aus Sicht des Deutschen Gewerkschaftsbundes (DGB) für die Beschäftigten nicht automatisch bessere Arbeitsbedingungen.

Im Gegenteil: Von den Arbeitnehmern, die in hohem oder sehr hohem Maß digitalisiert arbeiten, geben 46 Prozent an, dass ihre Arbeitsbelastung dadurch größer geworden ist. 45

DGB-Vorstandsmitglied Annelie Buntenbach

Prozent sehen keine Veränderung, und lediglich 9 Prozent fühlen sich durch die Digitalisierung entlastet. Die Zahlen, die der Deutschen Presse-Agentur vorliegen, seien erste Ergebnisse aus der repräsentativen Beschäftigtenbefragung zum DGB-Index »Gute Arbeit 2016«

mit Schwerpunkt »Digitalisierung der Arbeitswelt«.

»In der Debatte um Arbeiten 4.0 wird meist der Eindruck erweckt, als wären die Chancen und Risiken der Digitalisierung der Arbeitswelt gleich verteilt«, sagt DGB-Vorstandsmitglied Annelie Buntenbach der Deutschen Presse-Agentur. »Die aktuelle Befragung zeigt allerdings, dass digitales Arbeiten bislang für fast die Hälfte der Beschäftigten dazu führt, dass die Belastungen steigen.« Die höheren Belastungen seien eine Folge der Entgrenzung und Verlängerung der Arbeitszeiten, der permanenten Erreichbarkeit und der Arbeitsverdichtung.

Buntenbach verwies dabei auf ähnliche Ergebnisse einer Umfrage des Bundesarbeitsministeriums vom Januar, nach der zwei Drittel der Beschäftigten eine Intensivierung der Arbeit durch technologische Neuerung beklagen.

Buntenbach unterstrich: »Die Digitalisierung führt also nicht im Selbstlauf zu Entlastungen. Es braucht einen politischen Gestaltungsrahmen, damit der Trend, nach dem sich Arbeit zum Stressfaktor Nummer eins entwickelt hat, gedreht werden kann und durch die Digitalisierung nicht auch noch verstärkt wird. Es ist deshalb notwendig, dass die neuen Freiheiten durch Tablet und Smartphone ermöglicht, aber auch ausreichend abgesichert werden.«

Insbesondere neue Arbeitsformen wie mobiles Arbeiten und Homeoffice müssten politisch gestaltet werden, um – in der Regel unbezahlte – Überstunden und Stress zu vermeiden, mahnte Buntenbach. Im Rahmen des Dialogs des Arbeitsministeriums sollen auch Fragen flexiblerer Arbeitszeiten erörtert werden. Der übliche Acht-Stunden-Tag funktioniere nicht mehr in allen Branchen.

www.dgb.de

ownCloud

Eigener Marketplace für Apps

Mit dem eigenen Marketplace für Apps wird ownCloud das Ökosystem rund um ownCloud Server, Community-Apps und Enterprise-Apps erweitern.

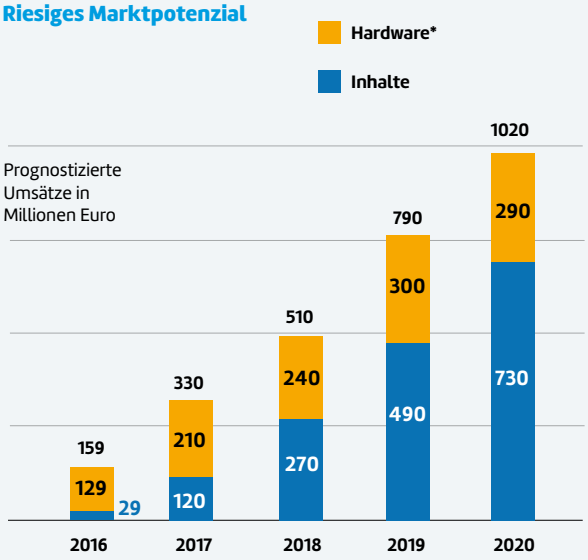
Entwickler und Partner können künftig den Marketplace nutzen, um ihre sowohl kommerziellen als auch Open-Source-Apps zu vermarkten. Sie können so Umsätze aus Lizenzen, Wartung und Support generieren oder die Apps kostenlos verteilen. Der ownCloud Marketplace wird verschiedene Lizenzmodelle unterstützen, um der Flexibilität moderner, Cloud-basierter IT-Infrastrukturen gerecht zu werden. »Die Möglichkeit zur Erweiterung und Anpassung von ownCloud gab es zwar schon immer, aufgrund fehlender Monetarisierungsmöglichkeiten wurde sie aber wenig genutzt«, erläutert ownCloud-CEO Tobias Gerlinger. »Entwickler aus der Community sind mit dem Wunsch auf uns zugekommen, eine Vermarktungsplattform zu schaffen. Der ownCloud Marketplace bietet Entwicklern und Partnern die Möglichkeit, ihre spannenden Erweiterungen und Apps zu vermarkten und zu verteilen. Ziel ist es, mit dem neuen Marketplace das Ökosystem von ownCloud weiter zu stärken und die führende Position von ownCloud am Markt auszubauen.«

www.owncloud.com/de

ownCloud will eine Vermarktungsplattform schaffen

Virtual Reality

Riesiges Marktpotenzial



Virtual Reality soll bis 2020 über eine Milliarde Euro umsetzen

web & mobile developer 11/2016
Quelle: Deloitte / *CE-relevante Hardware (Mid-Range, Full Feature), ohne Zubehör

Virtual Reality treibt Entwicklung der Unterhaltungselektronik an. Das zeigt die Trendstudie »Consumer Technology 2016« des Digitalverbands Bitkom und des Prüfungs- und Beratungsunternehmens Deloitte. »Viele Verbraucher sind im laufenden Jahr erstmals mit Virtual Reality in Berührung gekommen. Die Resonanz zeigt, dass die Technologie riesiges Potenzial hat«, sagt Timm Lutter, Bitkom-Experte für Consumer Electronics & Digital Media. So hat weit mehr als die Hälfte (59 Prozent) der Deutschen ab 14 Jahren schon von Virtual-Reality-Brillen gehört oder gelesen und fast jeder Dritte (31 Prozent) kann sich sogar vorstellen, eine VR-Brille zu nutzen. Fast jeder Zehnte (9 Prozent) hat bereits eine VR-Brille ausprobiert. Spezielle VR-Brillen, mit denen Verbraucher in virtuelle Realitäten eintauchen können, wurden von den Geräteherstellern in den vergangenen Monaten in Serie präsentiert oder angekündigt.

Laut der Trendstudie werden in Deutschland im Jahr 2016 voraussichtlich 158 Millionen Euro Umsatz mit Virtual Reality gemacht. Davon entfallen 129 Millionen Euro auf Hardware wie VR-Brillen und 29 Millionen Euro auf spezielle VR-Inhalte. Für das Jahr 2020 weist die Studie eine Prognose für Virtual-Reality-Umsätze im deutschen Markt in Höhe von rund einer Milliarde Euro aus. »Für eine neue Technologie ist dies ein auffallend starkes Wachstum«, sagt Klaus Böhm, Media Director bei Deloitte. »Virtual Reality wird sich zu einem zusätzlichen Standbein innerhalb der Consumer-Technology-Landschaft entwickeln.«

Mit Virtual Reality ist die Entwicklung noch nicht am Ende angekommen, denn die nächste Brillengeneration steht bereits in den Startlöchern: Sogenannte Mixed-Reality-Brillen, in denen die Realität durch holografische Bilder überlagert wird, sollen in den kommenden Jahren Marktreife erlangen.

www.bitkom.org

HackerRank

China hat die besten Programmierer

Laut einer Untersuchung von HackerRank hat China die besten Programmierer, gefolgt von Russland und Polen. Zum Vergleich: Die USA, die man gemeinhin als Coder-Nation schlechthin wahrnimmt, kommt gerade einmal auf Platz 28. Auch die Programmier-Großmacht Indien landet weit abgeschlagen auf Platz 31. HackerRank postet regelmäßig zu Trainingszwecken Programmieraufgaben und hat für seine Studie nun die Ergebnisse analysiert.

Schließlich hat HackerRank die Programmierkünste noch in diverse Sparten unterteilt. Daraus geht beispielsweise hervor, dass in der Schweiz die besten Datenbank-Programmierer zu finden sind, während die Programmierer aus Finnland in Sachen Ruby glänzen und in Hongkong offenbar Python die Programmiersprache der Wahl darstellt.

C++ scheint bei den Entwicklern in Frankreich hoch im Kurs zu stehen, und SQL ist eine Spezialität von Programmierern aus Dänemark. Selbst exotische Länder kommen zum Zuge: Laut HackerRank hat es den Programmierern in Sri Lanka das Thema Distributes Systems besonders angetan. Deutschland taucht in keinem Themengebiet unter den besten Fünf auf.

www.hackerrank.com

Countries With the Best Developers By Domain

(Ranked By Average Score)

Algorithms

Rank	Country
1	Russia
2	Poland
3	China
4	Switzerland
5	Taiwan

Java

Rank	Country
1	Poland
2	Bulgaria
3	Hungary
4	Switzerland
5	Russia

Data Structures

Rank	Country
1	China
2	Taiwan
3	Philippines
4	Russia
5	Japan

C++

Rank	Country
1	France
2	Russia
3	Hungary
4	Italy
5	Switzerland

Tutorials

Rank	Country
1	Hungary
2	Italy
3	Switzerland
4	Poland
5	Sweden

Mathematics

Rank	Country
1	China
2	Czech Republic
3	Finland
4	Ukraine
5	Japan

Python

Rank	Country
1	Hong Kong
2	Poland
3	Bulgaria
4	China
5	Russia

SQL

Rank	Country
1	Denmark
2	Finland
3	Switzerland
4	Russia
5	Hong Kong

Shell

Rank	Country
1	Czech Republic
2	Hong Kong
3	Hungary
4	Poland
5	China

Artificial Intelligence

Rank	Country
1	Japan
2	Belgium
3	Vietnam
4	Russia
5	Ireland

Functional Programming

Rank	Country
1	China
2	Switzerland
3	Russia
4	Japan
5	Taiwan

Databases

Rank	Country
1	Switzerland
2	Italy
3	Denmark
4	Taiwan
5	Hungary

Ruby

Rank	Country
1	Finland
2	Nigeria
3	Switzerland
4	France
5	Poland

Distributed Systems

Rank	Country
1	Sri Lanka
2	Malaysia
3	Pakistan
4	Ukraine
5	Finland

Security

Rank	Country
1	Ukraine
2	China
3	Switzerland
4	Czech Republic
5	Colombia

HackerRank

Die besten Entwickler nach Sprachen und Technologien

Sicherheit

Olympische Spiele 2016 im Fadenkreuz von Hackern

Arbor Networks Inc. berichtet im neuesten Blogeintrag des Sicherheitsteams ASERT (Arbor's Security Engineering & Response Team) über die außergewöhnlich starken Cyberangriffe auf die Olympischen Spiele 2016 in Rio de Janeiro.

Arbor verzeichnete während der Spiele die am längsten andauernden DDoS-Volumenangriffe, die jemals gemessen wurden. Die Angriffe konnten von den Sicherheitsteams abgewehrt werden, ohne dass die Spiele gestört wurden.

Bereits vor der Eröffnungszeremonie der Olympischen Spiele in Rio wurden öffentliche Webseiten der Olympischen Spiele und weitere beteiligte Unternehmensnetzwerke zum Ziel von andauernden, komplexen DDoS-Angriffen mit Spitzenvolumina bis zu 540 GBit/s. Einige der Angriffe begannen bereits Monate vor der Eröffnung der Spiele, nahmen aber während der Wettkämpfe weiter signifikant zu.

Die Angriffe erreichten ihren Höhepunkt in dem längsten je gemessenen Dauerangriff mit über 500 GBit/s, blieben aber von den Medien völlig unbeachtet und folgenlos für die Internetpräsenz der Olympischen Spiele.

Die Sicherheitsspezialisten Roland Dobbins & Kleber Carriello bei Arbor Networks enthüllen in einem aktuellen Beitrag des ASERT-Blogs Einzelheiten zu den spektakulären Angriffen und erfolgreichen Maßnahmen zur DDoS-Abwehr. Die Haupterkenntnisse der olympischen Rekord-Angriffe in der Übersicht:

Eine charakteristische Eigenschaft der gesamten IT-Sicherheitslandschaft, die auch für DDoS-Angriffe gilt, ist die Nutzung neuer Angriffsmethoden zuerst nur durch hochspezialisierte und kompetente Hacker. Häufig handelt es sich um Methoden, die zwar schon seit Jahren – manchmal Jahrzehnten – existieren, bisher aber nur sporadisch eingesetzt wurden. Diese Angriffsmethoden werden von den Angreifern permanent weiterentwickelt und erst viel später als einfach einzusetzende, automatisierte Angriffs-Tools auch für weniger oder nicht versierte Angreifer verfügbar gemacht.

Die Sicherheitsteams der olympischen Online-Präsenz wussten, dass sie zum Ziel von Angriffen werden würden, und bereiteten sich dementsprechend vor. Bereits vor Beginn der Spiele bewältigten die Sicherheitsteams einen Großteil der Arbeit zum Schutz der olympischen Internetpräsenz: Die gesamte Serverlandschaft mit all ihren Services, Applikationen und Zugangsmechanismen wurde untersucht, abgebildet und gescannt.

Dedizierte Anomalie-Erkennungsmechanismen wurden in Arbor SP eingespeist sowie spezielle situationsabhängige DDoS-Gegenmaßnahmen für Arbor TMS ausgewählt und konfiguriert.

Das Team der Arbor Cloud koordinierte seine Gegenmaßnahmen und Cloud-DDoS-Schutzmechanismen für eine lückenlose Abdeckung. Virtuelle Teams mit ausgewähltem Personal der beteiligten Unternehmen wurden gebildet, um die Verfügbarkeit der Netzwerk-Infrastruktur zu garantieren.

Best-Practice-DNS-Dienste wurden gewählt und implementiert, Kommunikationskanäle definiert sowie Prozeduren und Operationsmechanismen festgelegt und vieles mehr.

http://de.arbornetworks.com

NMG Business Solutions

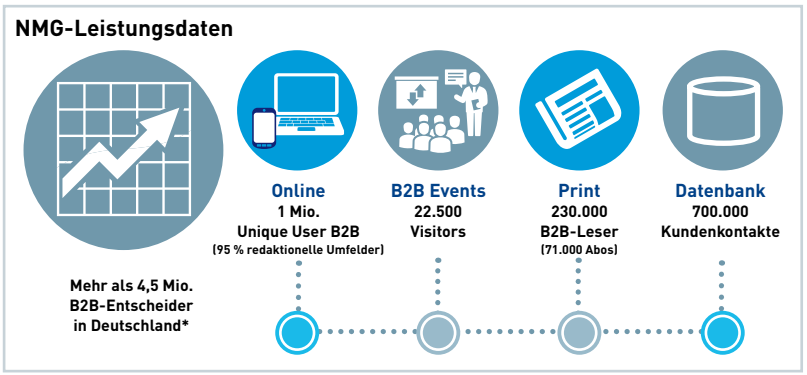
Leadgenerierung mit B2B Content Marketing

Wie kann man aus anonymen Zielgruppenkontakten wertvolle Multiplikatoren für die eigenen Produkte und Services gewinnen? Die Neue Mediengesellschaft Ulm bietet dafür eine Vielzahl von Kanälen

Die Verteilung der Marketingbudgets auf unterschiedliche Disziplinen ist eine große Herausforderung für den Marketingentscheider, denn er muss sich zunehmend vor anderen Instanzen im Unternehmen für die Wirksamkeit seiner Maßnahmen rechtfertigen.

Einer aktuellen Untersuchung zufolge rücken deshalb vor allem diejenigen Marketingaktivitäten in den Fokus, deren Messbarkeit in Form von Leads sichergestellt ist – ein Lead ist ein Kontakt mit einem aktiv interessierten potenziellen Kunden. Im Business-to-Business-Bereich sind die Leads am wertvollsten, die aus persönlichen Treffen auf Messen, Konferenzen und eigenen Veranstaltungen entstehen. Danach kommen die Leads, die mithilfe von digitalem Marketing entstehen: durch PR, Content Marketing und das klassische Direktmarketing. Dazu werden verschiedene Techniken eingesetzt, zum Beispiel Guided-Selling-Systeme, die potenzielle Käufer beraten und durch den Produktauswahlprozess leiten. All dies wird begleitet von einer immer tiefer gehenden Datenanalyse, die sich als „Smart Analytics“ zu einer eigenen Disziplin im Marketing entwickeln konnte.

Doch auch nachdem die grundlegende Verteilung der Budgets entschieden ist, müssen die Marketer darauf achten, ihre Maßnahmen intelligent zu orchestrieren. Denn der Erfolg der Aktivitäten hängt maßgeblich davon ab, ob die Entscheider und Mitentscheider (Beeinflusser) als Zielkunden durchgängig und nachhaltig überzeugt werden können. Damit dies gelingt, muss allen Marketingmaßnahmen ein Lead-Wert zugeschrieben werden, auch wenn die Medialeistung nicht nach „Konversion“ oder „Transaktion“ eingekauft werden kann. Kompetente erfahrene Mitarbeiter sind dafür wichtig – wo sie fehlen, bieten sich als Alternative eine Agentur oder ein Berater an.



*inkl. Ad Network ab Oktober 2016

Quellen: IVW, Agof, Google Analytics

Seit 2015 bietet sich die Neue Mediengesellschaft Ulm mbH (NMG) ihren Kunden im Projekt „Business Solutions“ als externer Partner an. Das Ziel: dem Kunden individuelle Konzepte und Lösungen für die oben beschriebenen Aufgaben zu bieten. Im Mittelpunkt steht der intelligente und direkte Weg zur Zielgruppe, unabhängig vom Medium (Event, Print, Digital). Gefragt ist ein professionelles Projektmanagement für maßgeschneiderte Lösungen. Ein alle Medien umfassendes CRM-System, crossmediales CMS und ein integriertes Data Mining wurden implementiert. Sie stehen allen Abteilungen des Hauses – und den Kunden – zur Verfügung.

In der Praxis ist es hilfreich, neben dem eigenen „Daten-Gold“ externe Datenquellen intelligent zu nutzen, unter Wahrung höchster datenschutzrechtlicher Vorgaben. An vielen Punkten ist dabei Technik im Spiel, auf den Einsatz von Menschen wird jedoch nicht verzichtet: Oft ist es einfacher, durch ein Telefonat fehlende Daten zu ergänzen oder Prozesse zu optimieren als Schnittstellen zu programmieren. Um etwa ein rechtssicheres Einverständnis zur Kontaktaufnahme einzuholen, hilft der ergänzende Einsatz von B2B-Telemarketing.

Die Technik und das intelligente Design der Organisation ist jedoch wenig wert ohne die entsprechende Durchschlagskraft in der Kommunikation: Hier schafft journalistische Qualität bei der NMG die notwendige Reichweite. Über 30 Fachjournalisten in München und Zürich werden ergänzt um Fachreferenten und Praktiker. Zudem wirken qualifizierte Producer und Developer für Online, Mobile, Video, Print, Seminare, Kongresse und Messen zusammen, um die Inhalte zur Zielgruppe zu transportieren.

Sie wünschen eine eigene individuelle Lösung? Informieren Sie sich unter **www.nmg.de/mediainfos** oder schreiben Sie direkt an unser Sales-Team: **sales@nmg.de**

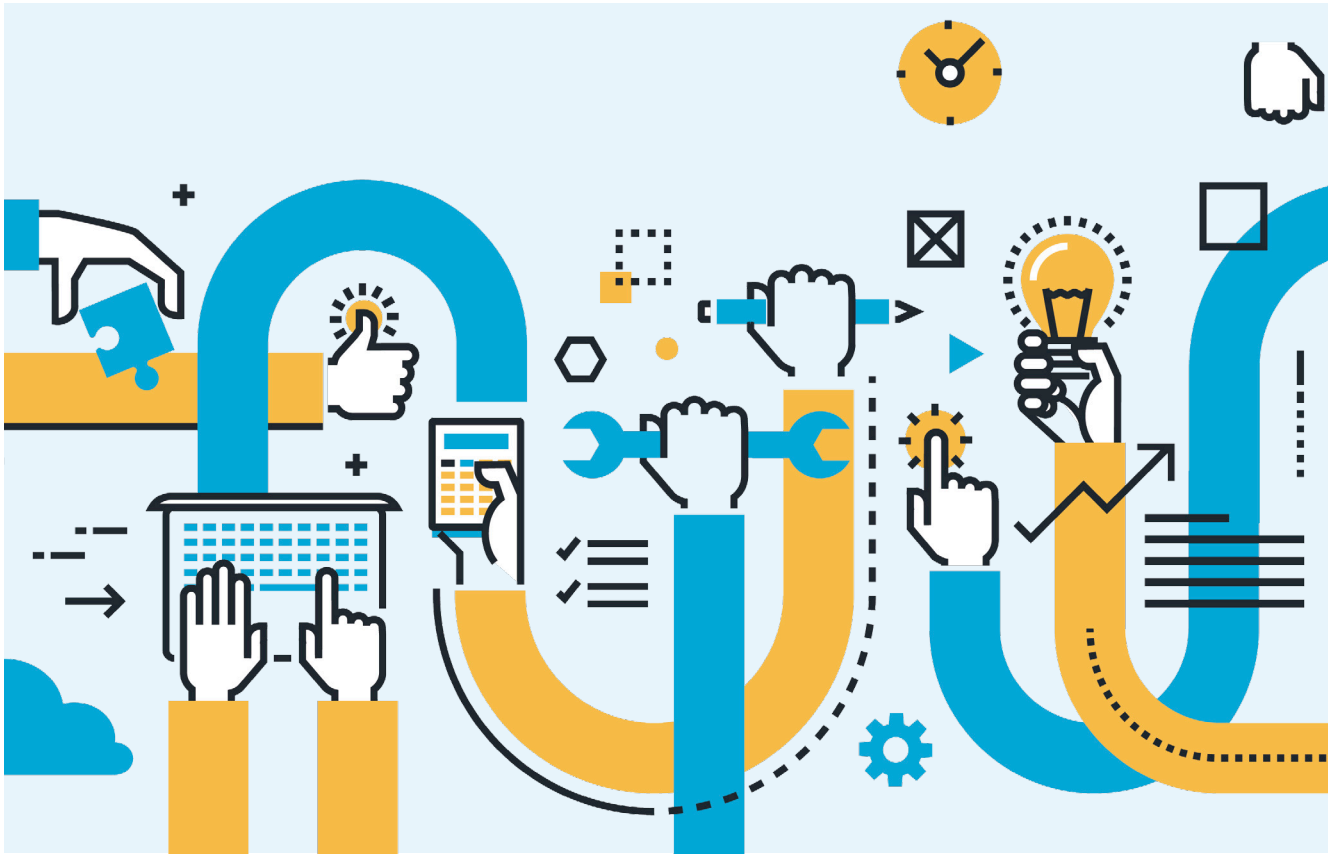


Bild: shutterstock / PureSolution

KOMPONENTENBASIERTE ARCHITEKTUREN

Mit Komponenten entwickeln

Als Webentwickler hört man häufiger denn je den Begriff Komponente.

Sei es beim Thema WebComponents, in dem viele die Zukunft des Webs sehen, in React Components oder neuerdings auch in Angular 2 Components: Der Begriff Komponente ist in aller Munde. Was hat es damit auf sich, und warum ist das wichtig? Welche Möglichkeiten haben Entwickler heute, mit Komponenten zu entwickeln?

Manche Begriffe der IT-Welt entziehen sich erfolgreich einer exakten Definition. Dazu gehört auch der Begriff Komponente. Nutzt man diesen Begriff ohne bessere Eingrenzung, so stößt man schnell auf Widerstand: Man wird darüber belehrt, dass dies ja eigentlich gar keine Komponenten im eigentlichen Sinne seien. In diesem Artikel geht es um komponentenbasierte UI-Architektur, insbesondere in einem Kontext von Webanwendungen.

Wenn es um Architekturen für Bedienoberflächen geht, hört man sehr oft die Abkürzung MVC – Model-View-Controller. Dieses Architekturmodell stammt noch aus Zeiten, in denen die Programmierung grafischer Bedienoberflächen in den Kinderschuhen steckte: Smalltalk bei Xerox-Parc.

Viele UI-Frameworks für Webanwendungen adaptierten das MVC-Konzept oder einen der unzähligen Abkömmlinge (beispielsweise MVVM). Als eine der wichtigsten Grundlagen galt stets »Separation of Concerns«. Dinge, die etwas anderes betreffen, sollten auch getrennt vom Rest definiert werden. Der Klassiker dafür ist die Trennung von Model und View: Die Behandlung der zugrunde liegenden Daten sollte nicht in den View-Code gepackt werden, und umgekehrt soll UI-Zustand (Auswahl, Ein-/Ausblendzustände) nicht in die Model-Daten gesteckt werden. Das Anwendungsverhalten sollte außerdem in Controllern definiert werden und hat weder in der View noch im Model etwas zu suchen.

Im Zuge dieser Konzepte entstand auch das Paradigma der objektorientierten Programmierung mit den Prinzipien Vererbung, Kapselung und Polymorphie. Jahrelang basierte die

Sprache, in der wir uns über die Bausteine zur Konstruktion von Bedienoberflächen austauschten, auf Begriffen wie View, Model und Controller, beschrieben durch objektorientierte Vererbungshierarchien. Meist wird versucht, neue Bibliotheken und Frameworks wie Backbone, Angular oder React in das bekannte, feste Schema dieser Terminologie zu pressen.

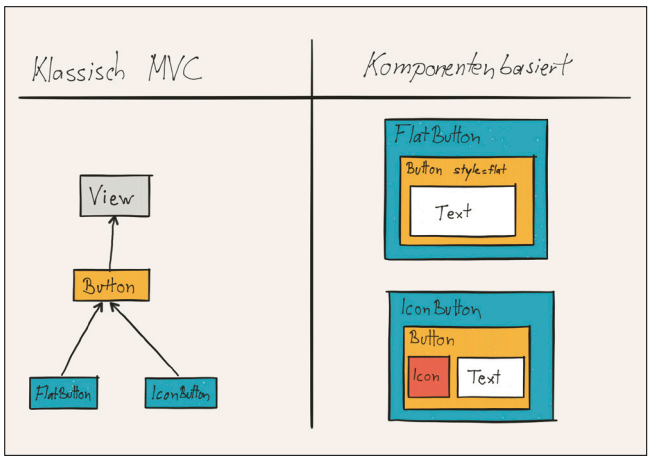
So findet man in der React-Dokumentation immer noch den Hinweis, dass manche Nutzer React als den View-Anteil ihrer Anwendung betrachten. Wer die Diskussionen der React-Entwickler verfolgt, wird feststellen, dass diese Auffassung sich sogar oft als kontraproduktiv erweist. Denn React setzt eine Architektur um, die man wesentlich sinnvoller als komponentenbasierte UI-Architektur auffasst und deren Grundidee eigentlich ganz anders ist.

Alles in Komponenten

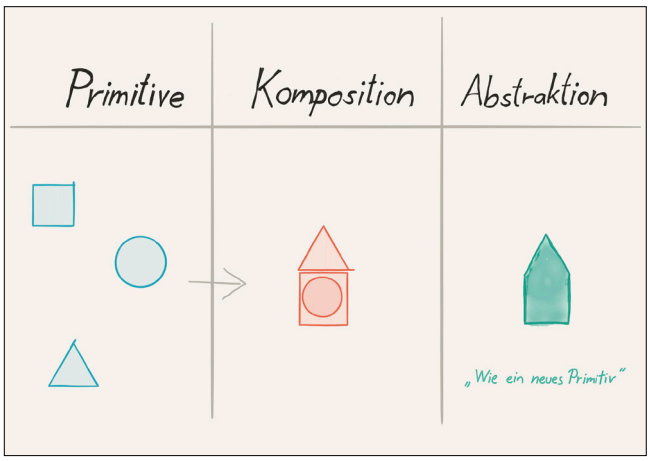
Heißt das, es gibt keine Views und Models oder Controller mehr? Nein – allerdings ist es nicht notwendig, diese drei Konzepte so hervorzuheben. In erster Linie ist jedes einzelne Ding einer Benutzerschnittstelle eine Komponente. Ganz egal ob es sich dabei um Fenster, Dialoge, Schaltflächen, Auswahlboxen, Kartendarstellungen und vieles mehr handelt. Ein Begriff für alles: Komponente.

Eine grundlegende Eigenschaft von Komponenten ist, dass sie miteinander kombinierbar sind. Man kann sich leicht vorstellen, wie man mehrere Schaltflächen, eine Textdarstellung und einen Dialog miteinander kombinieren kann. Komponenten sind die Legobausteine, aus denen man einfache bis komplexe Benutzerschnittstellen zusammenstellen kann.

Bild 1 verdeutlicht dies in einer Gegenüberstellung des klassischen MVC-Modells mit einem komponentenbasierten Ansatz: Bei MVC würde man FlatButton und IconButton typischerweise als Ableitung von einer Basis-View Button realisieren. Bei einem komponentenbasierten Ansatz verzichtet man auf die Vererbung. Stattdessen realisiert man einen FlatButton durch Komposition, indem man als Kindkomponente einen Button benutzt, der wiederum eine Text-Komponente für seine Beschriftung erhält. Ein IconButton folgt demselben Prinzip – auch dieser enthält eine Button-Komponente, aber



MVC versus komponentenbasiert: Architekturen im Vergleich (Bild 1)



Designsprache mit drei Aspekten: Primitive, Komposition und Abstraktion (Bild 2)

diese enthält nicht nur eine Text-Komponente als Beschriftung, sondern auch eine Icon-Komponente, die ein grafisches Symbol anzeigt.

Der Nutzen eines komponentenbasierten UI entsteht durch die Definition einer Designsprache, die es erlaubt, komplexe Oberflächen aus einfachen Bausteinen zu formulieren. Dabei gibt es drei Aspekte (Bild 2):

- **Primitive:** Die Primitive der Designsprache sind bei Weboberflächen die Standard-HTML-Elemente. Jedes *div, span, input* oder *table* stellt dabei eine konkrete Funktionalität als Baustein zur Verfügung.
- **Komposition:** Die Primitive können miteinander kombiniert werden. Bei HTML werden Elemente einfach ineinander verschachtelt.
- **Abstraktion:** Eine konkrete Komposition aus Primitiven kann einen Namen bekommen und unter diesem Namen einfacher und im Idealfall gekapselt wiederverwendet werden. Abstraktionen können genauso wie Primitive wieder kombiniert werden, sodass auch komplexere Abstraktionen entstehen können.

Wie die Beispiele zu diesen drei Aspekten verdeutlichen, sind die Elemente in HTML bereits nahezu so etwas wie eine komponentenbasierte Architektur. Elemente sind unabhängige Bausteine und lassen sich leicht in vielfältiger Weise miteinander kombinieren. Die Schnittstellen zwischen den Komponenten ergeben sich aus Verschachtelung und Eigenschaften dieser Elemente. Was jedoch noch fehlt, ist ein Mittel, um neue Abstraktionen zu schaffen.

Erste Versuche

Die Idee, abstrakte UI-Komponenten mit HTML zu realisieren, ist nicht neu: Eines der ersten und am weitesten verbreitete Frameworks für UI-Komponenten ist jQuery UI. Damit kann man Teilabschnitte des DOM, die einem bestimmten Aufbau folgen, per Aufruf einer JavaScript-Funktion zu einer UI-Komponente aufwerten. Kalender-Widgets, Buttons, Accordions, TabViews – die Zahl der UI-Komponenten wuchs schnell. ▶

Einen sehr ähnlichen Ansatz verfolgt das bekannte Framework Bootstrap. Es bietet nicht nur ein mit LESS realisiertes Grund-Layout mit klaren Vorgaben für Typografie und ein Layout-Gitter, sondern es bringt auch eine ganze Reihe von UI-Komponenten auf Basis von jQuery mit. Der Vorteil: Die Komponenten sind durch das CSS-System des Frameworks optisch flexibel anpassbar. Durch einfaches Austauschen eines Themes lässt sich das Aussehen der Komponenten weitgehend verändern.

All diese auf jQuery basierten Ansätze leiden unter dem Problem, dass die Interaktion zwischen einzelnen Komponenten einer komplexen Anwendung immer schwieriger wird. Eine Lösung für dieses Problem kann eine Bibliothek wie Backbone.js sein.

Diese kapselt die Komponenten und stellt eine besser abgeschlossene Abstraktionsebene bereit, die unabhängig vom HTML-DOM funktioniert. Die Idee dahinter ist, dass der Programmierer im Idealfall nur noch den State innerhalb von Backbone-Models modifiziert und die UI-Komponenten das DOM dann entsprechend dieses Zustands anpassen. Eine Modifikation des DOM außerhalb der *render()*-Methoden einer Backbone-View sollte vermieden werden.

W3C WebComponents kommen ins Spiel

WebComponents sind ein Konzept, das 2011 von Alex Russell auf der Fronteers Conference vorgestellt wurde. Alex Russell ist Entwickler bei Google und arbeitet unter anderem an Chrome, Blink und Googles Webplattform.

Die Idee zu WebComponents ist erstaunlich simpel. Der fehlende Baustein von HTML zu einer komponentenbasierten Architektur war die Möglichkeit, kombinierte Elemente zu einer neuen Komponente zusammenzustellen, die wieder wie ein Element verwendet werden kann. Mit WebComponents soll genau das möglich werden: Man kann selbst einfach einen neuen Element-Typ erfinden, der intern aus anderen Elementen aufgebaut ist.

Dazu wurde das Konzept in vier neue Webstandards aufgeteilt, die von Browserherstellern umgesetzt werden sollen:

- **Custom Elements:** Webentwickler sollen selbst eigene DOM-Elemente definieren können. Statt * … * soll man einfach *<x-calendar />* schreiben können.
- **HTML-Imports:** Es soll möglich sein, in HTML-Dokumente andere HTML-Dokumente mit sämtlichen Ressourcen zu importieren. Dies soll das Einbinden von WebComponents vereinfachen – denn diese bestehen ja möglicherweise aus einer HTML-Datei mit Templates, aus Stilen und Skript-Anteilen. Auch Bild-Ressourcen oder Zeichensätze können wiederum über die so verlinkten Inhalte indirekt referenziert werden.
- **Templates:** Mit Templates soll man in einem HTML-Dokument DOM-Teilbäume als Vorlage festlegen können. Diese werden nicht dargestellt, sondern dienen lediglich als Grundlage für neue Dokument-Fragmente.
- **Shadow DOM:** Das Shadow DOM ist neben den Custom Elements der wohl wichtigste Teil des WebComponents-Konzepts. Man kann damit die Elemente, aus denen die Implementierung eines Custom-Elements besteht, im herkömmlichen DOM (dem Light DOM) verstecken. Man sieht als Nutzer lediglich ein gewöhnliches Element. Erst indem man das Shadow DOM eines Elements öffnet, kann man den internen Aufbau erkennen. Dies soll zu einer stärkeren Kapselung von Webkomponenten führen.

Das Konzept sorgte für viel Aufsehen und viele sahen darin die große Zukunft des modernen Webs. Google veröffentlichte 2013 mit Polymer ein neues, auf WebComponents basierendes Framework, um das Konzept zu pushen. Damit soll es durch ein High-Level-API leichter werden, WebComponents zu implementieren.

Doch nach mittlerweile fünf Jahren ohne die erwartete Revolution fragen sich manche Entwickler, was aus der ganzen Idee geworden ist. Die Antwort: Die anderen Browser-Hersteller haben letztlich nicht so mitgezogen wie gehofft. Die einzige halbwegs funktionierende Implementierung befindet sich in Chrome. Google ist bei Entwurf und Implementierung weitgehend ohne Mitwirkung der Mitbewerber vorausgestürmt – man wollte eben nicht warten, bis alle mitziehen. Mangels Mitwirkungsmöglichkeiten gab es verständlicherweise ein geringeres Interesse anderer Entwicklerteams.

Doch glücklicherweise hat sich das mittlerweile geändert. Templates werden immerhin bereits umfangreich unterstützt und Safari (WebKit) implementiert bereits die überarbeitete Fassung der Shadow-DOM-Spezifikation. Allerdings treten jetzt immer mehr Schwächen und Inkonsistenzen zutage, die bislang nicht aufgefallen sind. Dies führte vor allem in letzter Zeit zu teilweise größeren Änderungen dieses immerhin seit fünf Jahren im Netz herumgeisternden API. Besonders betroffen ist davon momentan die Custom-Elements-Spezifikation. Listing 1 zeigt ein Beispiel einer Custom-Element-Definition mit dem bislang standardisierten API.

Dieser Code entspricht zwar dem bislang verabschiedeten Standard und funktioniert so auch in Google Chrome – allerdings ist er gegenüber den bereits weiterentwickelten Spezifikationen veraltet. Die anderen Webbrowser werden mit hoher Wahrscheinlichkeit ein anderes API implementieren.

Als ES2015-Klasse implementiert

Das Beispiel definiert ein neues Custom-Element mit dem Namen *x-slideshow*. Das Element ist als ES2015-Klasse implementiert und leitet von *HTMLElement* ab. Das muss nicht so sein – man kann auch mit Prototypen arbeiten. Der *createdCallback* wird aufgerufen, wenn ein Custom-Element erzeugt wird. Darin initialisiere ich die aktuelle Slide der Slideshow und erzeuge mit *this.createShadowRoot()* ein Shadow DOM für das Custom-Element.

In einem solchen Shadow DOM kann man Elemente einbetten, die im normalen Browser-DOM unsichtbar sein sollen. Der Inhalt des Shadow DOM wird im Beispiel mit einer einfachen Template-Zeichenkette initialisiert. In dieser Vorlage befindet sich ein *style*-Block. Dessen Stile sind automatisch innerhalb des Shadow DOM gekapselt. Weiterhin befindet sich darin ein Wrapper-*div*-Element, das ein *content*-Element umschließt. Mit dem *content*-Element kann man aus ei-

nem Shadow DOM heraus Elemente im Light DOM selektieren. Im Beispiel werden so die Kinder des Custom-Elements in das Wrapper-DIV des Shadow DOM gezogen. Am Ende des *createCallback* wird noch mittels *updateSlider* die aktuelle Slide gewählt, indem alle anderen Kinder bis auf das aktuelle versteckt werden. Mit einem Intervall-Timer wird der Index der aktuellen Slides stets weitergeschaltet und die Kinder mit *updateSlider* dazu passend ein- und ausgeblendet.

Die Funktion *document.registerElement()* dient zum Definieren eines Custom-Elements. Allerdings wird hier eigentlich kein Element mit irgendetwas registriert, sondern ein Custom-Element definiert. Deshalb wurde die Methode zwischenzeitlich in *document.defineElement* umbenannt. Doch schon bald kam es zu weiteren Einwänden. Unter anderem ging es um das Konzept des Element-Upgrading. Denn da Custom-Element-Definitionen ja möglicherweise erst später geladen werden als die Inhalte, können bereits Elemente erzeugt worden sein, die zwar den Namen eines Custom-Elements haben, womit aber der Browser noch nichts anfangen konnte. Man benötigt also einen Prozess, bei dem zuerst alle HTML-Elemente mit unbekanntem Tagnamen gesammelt werden, um sie dann bei der Definition eines Custom-Elements in ein solches Element umzuwandeln. Genau so wurde das auch in der bisherigen Spezifikation gehandhabt. Der WebKit-Entwickler Ryosuke Niwa postete jedoch vor einiger Zeit folgendes Codebeispiel:

```
for (var i = 0; i < 1000000; i++)
  document.createElement('my-element');
```

Diese Schleife würde eine große Zahl von Elementen mit dem Tagnamen *my-element* erzeugen, die dann alle für ein Up- ▶

Listing 1: Slideshow nach dem bisherigen WebComponents-API

```
'use strict'
const template = `
<style>
  .Slideshow {
    border: 1px solid gray;
  }
</style>
<div class="Slideshow">
  <content select="*"></content>
</div>
`

class Slideshow extends HTMLElement {

  createdCallback () {
    this._currentSlide = 0
    this._delay = parseInt
    (this.getAttribute("delay")
      || "1000")

    this._shadowRoot = this.createShadowRoot()
    this._shadowRoot.innerHTML = template

    this._updateSlider()
  }

  attachedCallback () {
    this._start()
  }

  detachedCallback () {
    this._stop()
  }

  _start () {
    if (!this._timer) {
      this._timer = setInterval(function () {
        this._currentSlide =
        (this._currentSlide + 1) %
        this.children.length
        this._updateSlider()
      }.bind(this), this._delay)
    }
  }

  _stop () {
    if (this._timer) {
      clearInterval(this._timer)
    }
  }

  _updateSlider () {
    let slide = [].slice.call(this.children)
    slide.map((child,ix) => {
      if (ix === this._currentSlide) {
        this._show(child)
      } else {
        this._hide(child)
      }
    })
  }

  _hide (el) {
    el.style.display = 'none'
  }

  _show (el) {
    el.style.display = 'block'
  }
}

document.registerElement('x-slideshow', Slideshow)
```

grade vorgemerkt werden müssten, obwohl sie vielleicht niemals in ein DOM eingefügt werden würden. Das kann zu einem Memory Leak führen. Nach einigem Hin und Her wurde Niwas Vorschlag angenommen: Man würde das automatische Upgrade von Elementen erst beim Einfügen in das DOM durchführen und nicht schon, wenn ein Element instanziert wird. Doch dazu waren weitere Änderungen am Standard notwendig. Weiterhin gibt es aktuell Überlegungen zu expliziten Methoden, die es ermöglichen, ein Element auch vor dem Einfügen in das DOM manuell zu upgraden – aber das ist auch noch unfertig.

Man muss damit rechnen, dass in nächster Zeit noch eine ganze Reihe von Methoden zu Custom Elements ergänzt werden. Die Mitglieder der Working Group haben sich deshalb überlegt, diese unter einem gemeinsamen Objekt *window.customElements* zu sammeln. Anstelle des zwischenzeitlichen *document.defineElement* soll es dann in Zukunft *window.customElements.define* lauten.

Eine ständig wiederkehrende Herausforderung ist das grundsätzlich sehr imperative API des DOM: Bei imperativem Code spielt Zeit eine wichtige Rolle: Elemente können jederzeit destruktiv in alte DOM-Strukturen eingefügt werden. Der Ladezeitpunkt von Custom-Element-Definitionen ist oft entscheidend. Voneinander abhängige Definitionen zusammen mit unterschiedlichen Ladezeitpunkten, Lazy Loading, Asynchronität und veränderlichen DOM-Strukturen macht es sichtlich schwer, derartige Dinge zu spezifizieren.

Listing 2 zeigt die momentan übliche Fassung für Custom Elements im Rahmen der WebComponents-Spezifikationen. Der *createCallback* wurde abgeschafft – stattdessen soll es

Listing 2: Slideshow nach dem neueren WebComponents-API

```
'use strict'

const template = `
<style>
  .Slideshow {
    border: 1px solid gray;
  }
</style>
<div class="Slideshow">
  <slot />
</div>
`

class Slideshow extends HTMLElement {

  constructor () {
    this._currentSlide = 0
    this._delay = parseInt
    (this.getAttribute("delay")
      || "1000")
    this._shadowRoot = this.attachShadow()
    this._shadowRoot.innerHTML = template
  }

  connectedCallback () {
    this._updateSlider()
  }

  attachedCallback () {
    this._start()
  }

  detachedCallback () {
    this._stop()
  }

  _start () {
    if (!this._timer) {
      this._timer = setInterval(function () {
        this._currentSlide =
          (this._currentSlide+1) %
          this.children.length
        this._updateSlider()
      }.bind(this), this._delay)
    }
  }

  _stop () {
    if (this._timer) {
      clearInterval(this._timer)
    }
  }

  _updateSlider () {
    let slide = [].slice.call(this.children)
    slide.map((child,ix) => {
      if (ix === this._currentSlide) {
        this._show(child)
      } else {
        this._hide(child)
      }
    })
  }

  _hide (el) {
    el.style.display = 'none'
  }

  _show (el) {
    el.style.display = 'block'
  }
}

window.customElements.define('x-slideshow', Slideshow)
```

Listing 3: Slideshow.js (React)

```
'use strict'
import React, {Component, Children} from 'react'

export default class Slideshow extends Component {
  state = {
    currentSlide: 0
  };

  componentDidMount() {
    this.startTimer()
  }

  componentWillUnmount() {
    this.stopTimer()
  }

  componentWillReceiveProps(newProps, newState) {
    if (newProps.delay !== this.props.delay) {
      this.stopTimer()
      this.startTimer()
    }
  }

  render () {
    let children = this.props.children
    let currentChild =
       Children.toArray(children)
         [this.state.currentSlide]
    return (
      <div className="Slideshow">
      {currentChild}
      </div>
    )
  }

  startTimer () {
    if (!this._timer) {
      let count =
      Children.count(this.props.children)
      this._timer = setInterval(function () {
        this.setState({
          currentSlide:
          (this.state.currentSlide+1)%count})
      }.bind(this), this.props.delay);
    }
  }

  stopTimer () {
    if (this._timer) {
      clearInterval(this._timer)
      this._timer = null
    }
  }
}
```

nun den herkömmlichen Konstruktur und zusätzlich einen *connectedCallback* geben, der im Rahmen des Upgrade-Prozesses aufgerufen wird. Statt *this.createShadowRoot()* müsste man nun *this.attachShadow()* aufrufen, um eine Shadow-DOM-Wurzel im Element zu erzeugen. All diese Änderungen sind hochaktuell und noch in keinem aktuellen Browser nativ implementiert. Wer bislang auf die alten APIs gesetzt hat, wird jedoch darauf bestehenden Code umschreiben müssen.

React Components

Im selben Jahr, in dem Alex Russell seinen Vortrag über Web-Components hielt, entwickelte der Facebook-Entwickler Jordan Walke die JavaScript-Bibliothek React und setzte sie im Facebook-Newsfeed ein. Ein Jahr später kam die neue Bibliothek – durch das Engagement von Pete Hunt – auch bei Instagram.com zum Einsatz. Im Mai 2013 veröffentlichte Facebook schließlich auf der JSConf-US React als neues Open-Source-Projekt und damit einen alternativen Ansatz für komponentenbasierte UIs. Dabei wurde React jedoch nie wirklich als Alternative zu den W3C WebComponents beworben, sondern entstand letztlich aus den technischen Anforderungen komplexer Webanwendungen, wie sie bei Facebook oder Instagram das Tagesgeschäft sind.

Weder Polyfills noch neue Webstandards sind für Reacts Komponentenmodell notwendig: Die React-Entwickler bedienen sich eines Kunstgriffs: Sie implementieren ein eigenes, virtuelles DOM, das durch einen DOM-Diff-Algorithmus mit dem realen DOM abgeglichen wird. Innerhalb dieses virtuellen DOM kann man mit React-Components neue Elementtypen implementieren. Da Facebook mit React ein eigenes DOM von Grund auf definiert, haben sie ein modernes, funktionales API geschaffen. Damit leidet dieses API auch nicht unter den Problemen, die das mit imperativen Altlasten ausgestattete W3C-WebComponents-API kennzeichnen.

Eine zur obigen WebComponent weitgehend äquivalente React-Komponente zeigt Listing 3.

Zum Vergleich: Auch die React-Komponente kann als Klasse definiert werden. Für das Einhängen und Entfernen aus dem DOM gibt es ebenfalls Methoden: *componentDidMount* entspricht *attachedCallback* und *componentWillUnmount* ist in etwa mit dem *detachedCallback* vergleichbar. Auch für den *attributeChangedCallback* gibt es eine Parallele: *componentWillReceiveProps* wird aufgerufen, wenn sich die Props der Komponente ändern.

Statt mit CSS-Eigenschaften wie *display* die einzelnen Slides ein- und auszublenden, löst die React-Komponente das Problem auf ungewöhnliche Weise: Es wird jeweils nur das gerade aktuelle Kind gerendert. React und der DOM-Diff-Algorithmus kümmern sich um ein effizientes Update des realen DOM. ▶

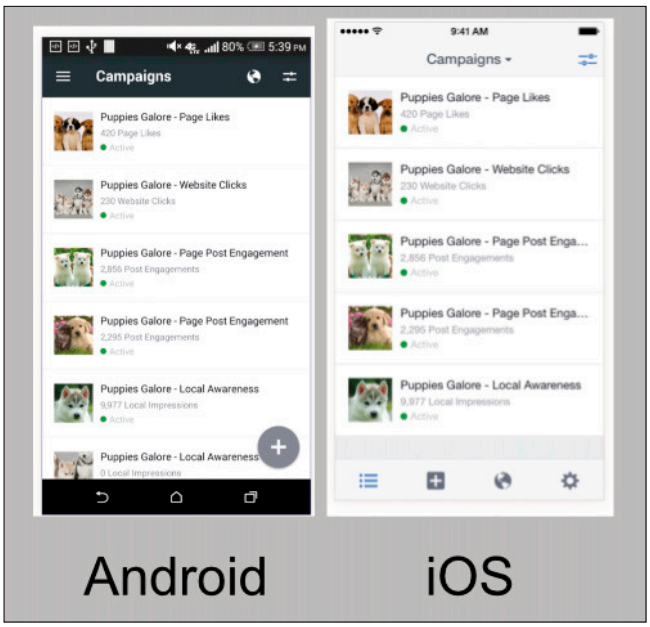
Doch ein Unterschied ist entscheidend: Auch wenn Komponenten als Klassen und Instanzen realisiert werden und sogar veränderlichen Zustand besitzen können: Das Virtual DOM in React ist eine unveränderliche Datenstruktur. Wenn sich die Props oder der interne Zustand einer Komponente ändern, dann wird das virtuelle DOM neu generiert.

Dies führt so weit, dass React eigentlich für Komponenten ohne Zustand nicht einmal Instanzen benötigt. Man kann Komponenten deshalb auch einfach als Funktion definieren:

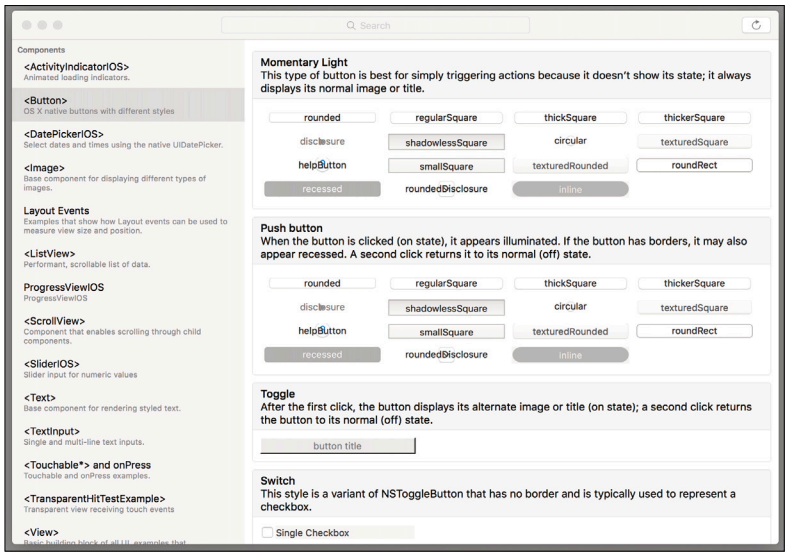
```
import React from 'react'
export default function Headline (props) {
  return (
    <h1 className="Headline">
    {this.props.children}
    </h1>
  )
}
```

Die Funktion entspricht dabei dann einfach dem, was in der Klasse die *render()*-Methode ist. Da es in der Funktion keine Instanz der Komponente gibt (also kein *this)*, werden die *props* einfach als Parameter übergeben. Derartige Komponenten werden Stateless Functional Components genannt. Bei diesen Funktionskomponenten kann man zwar die Lifecycle-Methoden nicht definieren, dafür lassen sie sich hervorragend mit funktionalen Programmiermustern einsetzen:

```
import {map} from 'lodash/fp'
let List = ({children}) => <ul
className="List">{children}</ul>
let UserItem = ({user}) => <li>User: {user.name}</li>
```



React Native für iOS und Android (Bild 3)



React Native for macOS: macOS-Anwendungen mit React (Bild 4)

```
let UserList = ({users}) => compose(List, map(Item))
render(<UserList users={getUsers()} />, document.body)
```

Die simple Verwendung eines so grundlegenden funktionalen Operators wie *compose* zeigt hier Composability in Reinform. Schon der Name macht es deutlich und es zeigt sich: Funktionen sind eine nahezu ideale Repräsentation für Komponenten, und die funktionale Programmierung beschäftigt sich seit Jahrzehnten genau damit, wie man Funktionen miteinander kombiniert und daraus neue Funktionen als Abstraktion erschafft. Genau das ist, was ein flexibles Komponentensystem ausmacht, und genau das macht React so flexibel und erfolgreich.

Eine häufig in React-Foren zu lesende Frage ist, wie das React-Projekt eigentlich zum Thema W3C WebComponents stehe. Denn spätestens wenn diese wirklich in den Webbrowsern umgesetzt sind, wäre es ja sinnvoll, darauf aufzubauen. Manch einer hält dann gar React insgesamt für obsolet. Die offizielle Haltung einiger der React-Kernentwickler ist jedoch sehr klar: WebComponents bieten laut ihnen keine Lösungen für die Probleme, denen sie sich stellen. Sie bezweifeln, dass WebComponents wirklich bei der Interoperabilität zwischen Frameworks helfen. Das API ist zudem imperativ ausgelegt und verträgt sich daher nur bedingt mit dem funktionalen Modell, das man von React kennt und schätzt. Doch ein ein anderer Aspekt ist noch wesentlich gewichtiger: Das Komponentensystem von React ist unabhängig von Webtechnologien definiert. Die vorgefertigten Komponenten für Webentwicklung befinden sich im Package *React.DOM*. Es gibt jedoch keine Abhängigkeiten des Komponentensystems zum Web oder dem HTML-DOM.

React Native, React Desktop und mehr

Aufgrund dieser Eigenschaft haben sich mittlerweile eine ganze Reihe von Projekten herausgebildet – allen voran das wiederum von Facebook initiierte Projekt React Native. Damit können Entwickler mittels React-Komponenten native

Anwendungen für iOS, Android und Windows 10 erzeugen. Die Komponenten werden nicht auf HTML-Elemente abgebildet, sondern auf native Views der jeweiligen Plattform (Bild 3). Mit dem Projekt React Desktop kann man macOS-Anwendungen entwickeln – wiederum mit React-Komponenten, die auf native Views abgebildet werden (Bild 4).

Ein besonders kurioser Renderer für React-Komponenten ist auch React Blessed: Die Komponenten werden dabei auf Konstrukte der Konsolenbibliothek blessed abgebildet. Man kann also auf diese Weise interaktive Konsolenanwendungen mittels React implementieren (Bild 5).

Ein essenzieller Unterschied zwischen React und WebComponents ist also, dass Letztere ausschließlich auf Webanwendungen ausgelegt sind und in dieses Umfeld integriert sind. Das Komponentenmodell von React ist allgemeiner und lässt sich auf nahezu beliebige Anwendungsfälle adaptieren.

WebComponents in React nutzen

W3C WebComponents als neuer Standard und React als Bibliothek mit einem eigenen Konzept für Komponenten schließen sich nicht gegenseitig aus. Man kann beide Technologien auch miteinander kombinieren. Dabei sind zwei Anwendungsfälle möglich: Man kann WebComponents in einer React Component so verwenden, wie man auch alle anderen HTML-Elemente verwenden kann:

```
function Gallery (props) {
  return (
    <div className="Gallery">
      <x-slideshow delay="1000">
        {props.children}
      </x-slideshow>
    </div>
  )
}
```

Wichtig ist dabei, dass die so eingesetzte WebComponent mit Kleinbuchstaben geschrieben wird und den obligatorischen Bindestrich aufweist. Hier zeigt sich deutlich der Sinn und Zweck von WebComponents: Bei der Nutzung unterscheiden sie sich im Idealfall nicht von eingebauten Elementen. Die Browserhersteller können sogar dazu übergehen, bestimmte Standardelemente einfach als WebComponent zu implementieren.

Die praktischen Schwierigkeiten liegen dabei nach wie vor im mangelnden Browsersupport für WebComponents begründet. Auch die sich ändernden Standards stellen ein Problem dar. So hatte React bereits Unterstützung für Ereignis-Handling in Shadow DOMs – allerdings noch in der alten Version des Standards.

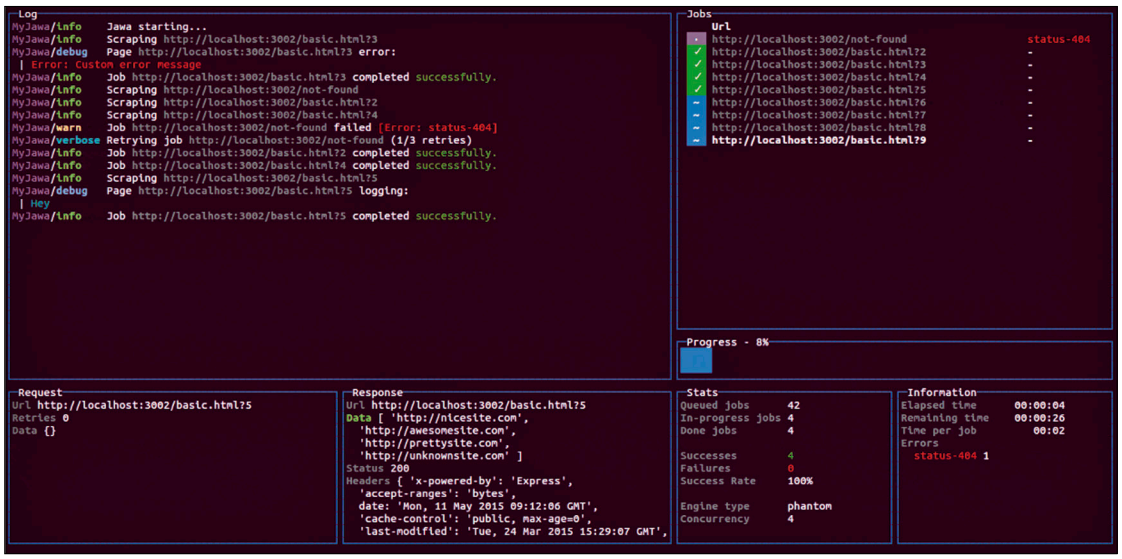
Ein anderes, grundsätzlicheres Problem sind die Übergänge zwischen virtuellem DOM und nativem DOM: Nutzt man eine WebComponent in React, dann entstehen die Kinder der WebComponent aus gerenderten React-Komponenten. Wenn die WebComponent diese Kinder manipuliert, kann dies zu Problemen in React führen. WebComponents mit Seiteneffekten sind keine gute Idee.

React-Komponente in WebComponent wrappen

Der zweite Anwendungsfall ist die Implementierung einer WebComponent mittels React. Der Standard für WebComponents sieht keine besonderen Mittel vor, wie man das interne DOM einer WebComponent konstruiert. Eine Möglichkeit sind natürlich die Templates aus den WebComponents-Spezifikationen. Das ist jedoch nicht zwingend.

Entwickler können auch einfach das herkömmliche DOM-API benutzen. Da dieses API aber mitunter unbequem für komplexe Dinge ist, haben sich diverse Bibliotheken als Erleichterung herausgebildet. Man kann also beispielsweise auch jQuery einsetzen, um die notwendige DOM-Manipulation in der WebComponent zu implementieren. Ebenso kann man React nutzen. Dazu definiert man eine React-Komponente und rendert diese dann in das Shadow DOM eines Custom-Elements (Listing 4).

Auch hier ist der Übergang von der Komponente auf ihre Kinder der eigentlich schwierige Aspekt. Denn im Cus- ▶



React Blessed: Interaktive Konsolenanwendungen mit React (Bild 5)

tom-Element sind die Kind-Knoten native DOM-Knoten. Diese können nicht direkt als Kinder einer React-Komponente benutzt werden. Verschiedene Strategien sind jedoch möglich, um diese Problematik zu umgehen:

- Man kann die nativen Elemente per *ref* einhängen. Dazu holt man sich in der React-Komponente eine Referenz auf das native Element und hängt dort dann die nativen Kindknoten ein.
- Man kann den textuellen Inhalt auslesen und in der React-Komponente rendern.
- Man baut aus dem nativen Teil-DOM ein virtuelles DOM, ähnlich wie dies preact-markup mit seinem HTML-Parser macht.

Ganz gleich welche Variante man wählt: Durch den Hin- und Herwechsel zwischen den Welten ist der Übergang nicht einfach und transparent.

SkateJS als Alternative

Die W3C-WebComponents-API ist imperativ ausgelegt. React stellt ein auf funktionalen Prinzipien aufbauendes Komponentensystem auf, nutzt dabei jedoch nicht die W3C WebComponents. Spätestens seit dem Erfolg von React fällt auf, dass funktionale Programmierung eine bessere Grundlage für komponentenorientierte UI-Programmierung darstellt. Sobald man vermeidet, Komponenten imperativ zu modifizieren, und sie stattdessen lediglich erzeugt und transformiert, verschwinden ganze Kategorien von Fehlern. Komponenten werden robuster und deren Verhalten ist einfacher nachvollziehbar. Sie hängen im Idealfall einfach nur von ihren Parametern ab.

So kann man bislang wählen zwischen einem Ansatz, der womöglich in zukünftigen Browsern zum Standard wird (W3C WebComponents), aber nicht die Vorteile funktionaler Programmierung bietet, oder einem, der bereits heute in praktisch allen Browsern funktioniert und der funktional orientiert ist, aber bei dem die Integration mit W3C WebComponents noch aufwendiger ist.

Die noch recht unbekannte kleine Bibliothek SkateJS soll eine Alternative dazu bieten: Sie baut direkt auf W3C WebComponents auf – Browser ohne entsprechende Unterstützung benötigen deshalb Polyfills.

Allerdings stellt SkateJS dann ein API bereit, mit dem Komponenten auf funktionale Art und Weise definiert werden können. Tatsächlich sogar sehr ähnlich, wie man das auch aus React kennt. Es verwundert dabei nicht, dass auch SkateJS auf ein virtuelles DOM aufbaut, um die funktional konstruierten DOM-Bäume auf das native DOM anzuwenden. Die Installation erfolgt wie üblich per NPM:

```
npm install skatejs  --save
```

Wer gleich ein passendes Set an Polyfills installieren möchte, kann dafür das Paket *skatejs-web-components* installieren:

```
npm install skatejs-web-components --save
```

Listing 5 implementiert als Beispiel eine kleine Zähler-Komponente mit Hilfe der Bibliothek SkateJS. Das sieht tatsächlich ein wenig wie eine Mischung aus dem neueren WebComponents-API und React aus.

Angular 2 setzt auf auf ES2015-Module

Der namentliche Nachfolger des bekannten Webframeworks Angular sorgte schon vor seinem Erscheinen für viel Unruhe in Entwicklerkreisen. Angular 2 wurde durch das Entwicklerteam derart radikal umdefiniert, dass viele das Ergebnis nicht wiedererkannten. Dabei waren viele der Änderungen durchaus naheliegend.

Angular als Framework hat praktisch alles angeboten, was man als Webentwickler brauchen kann. So gab es auch ein eigenes Modulsystem, obwohl sich schon damals mit CommonJS und AMD Quasistandards zu etablieren begannen. Derartige Eigenentwicklungen sind jedoch bei großen Frameworks nicht unüblich. Auch das bekannte Framework ExtJS von Sencha bot nahezu alles, was es in der Webentwickler-Welt gab, noch einmal in einer eigenen Fassung an.

Angular 2 ist dagegen in erster Linie eine starke Vereinfachung: Statt auf ein eigenes Modulsystem setzt man auf ES2015-Module. Statt mit Controllern und Direktiven arbeitet man in Angular 2 praktisch ausschließlich mit Komponenten. Auch hier sieht man, dass der Trend eindeutig in Richtung komponentenbasierte Entwicklung geht. Eine Komponente in Angular 2 ist eine Direktive (ähnlich wie eine Cus-

Listing 4: WebComponent intern mit React-Komponente implementieren

```
function HeadLine (props) {
  return (<h1 class="HeadLine">{ props.children }
  </h1>)
}

function getProps (el) {
  return { children: [el.textContent]}
}

class XHeadLine extends HTMLElement {
  attachedCallback () {

    let root = document.createElement('span')
    this.createShadowRoot().appendChild(root)
    let props = getProps(this)
    render(<HeadLine {...props} />, root)
  }
}

document.registerElement('x-headline', XHeadLine)
```

Listing 5: hello-counter.js

```
import 'skatejs-web-components'
import {define, vdom, prop} from 'skatejs'

define('x-hello-counter', {
  props: {
    hellos: prop.number({
      attribute: true,
      default: 0
    }),
    play: prop.boolean({
      attribute: true,
      default: true
    })
  },
  attached (el) {
    el.__interval = setInterval(()=>{
      el.hellos = el.hellos + 1}, 1000)
    })
  },
  detached (el) {
    clearInterval(el.__interval)
  },
  render (el) {
    vdom.element("div", {class: "HelloCounter"},
    vdom.text("Hellos: "),
    vdom.text(el.hellos))
  }
})
```

tom-Element-Definition) mit einem zugehörigen Template, das die Struktur beschreibt:

```
import {Component} from angular2/core

@Component({
  selector: "hello",
  template: '<div class="Hello">{name}</div>'
})

export class HelloComponent {
  name: string
  constructor(name:string) {
    this.name = name
  }
}
```

Eine weitere Änderung gegenüber Angular 1 ist die starke Orientierung an Microsofts typisierter Variante von JavaScript namens TypeScript. Angular 2 selbst ist in TypeScript implementiert und viele Tutorials legen nahe, dass man Angular-2-Anwendungen auch mit dieser Sprache implementieren soll. Im Vergleich zu React orientiert sich Angular 2 wesentlich deutlicher am herkömmlichen objektorientierten Programmierstil. Geschäftslogik soll weitgehend in sogenannten Services gekapselt werden, die dann von den Komponenten genutzt werden können. Doch nicht alles hat sich gegenüber dem Vorgänger geändert: Auch Angular 2 setzt in seinen textuellen Templates auf eine Spezialsyntax. Nicht nur die Interpolation von Variablen, sondern auch die Bindung von Ereignis-Handlern ist durch Spezialsyntax gelöst.

Hier unterscheidet sich Angular 2 sowohl vom WebComponents-Ansatz als auch von React. WebComponents besitzen keinerlei Spezialsyntax, und das JSX in React ist nur syntaktischer Zucker für Funktionsaufrufe.

Fazit

Komponentenbasierte UIs sind eine Lösung für komplexe Webanwendungen. Mittels einer durchgängigen Zerlegung der Anwendung in kleine, wiederverwendbare Komponenten wird das Resultat besser pflegbar, erweiterbar und präsentiert sich einheitlicher. Die Denkweise des Entwicklers wandelt sich dabei vom klassischen Top-down-Ansatz zu einem Bottom-up-Ansatz bei dem zuerst über die Bausteine nachgedacht wird, die man eigentlich bräuchte, um ein Problem leicht lösen zu können.

WebComponents sind ein spannender Ansatz für ein erweiterbares Web. Sie machen aus HTML selbst eine Komponenten-Architektur. Die praktische Anwendbarkeit ist noch begrenzt, da der Browsersupport noch sehr schlecht aussieht und die Standards noch im Fluss sind. Wer heute schon auf diese Weise arbeiten möchte, kann sich das vielversprechende Projekt SkaterJS ansehen.

React ist eine solide und beliebte Bibliothek. Das darin realisierte Komponentenmodell besitzt den Charme, auch über den Anwendungsfall Web hinaus einsetzbar zu sein. Nicht umsonst gibt es eine Vielzahl von Renderern wie React Native, React Desktop, React D3 oder React Blessed. Wer React erlernt hat, kann sein Wissen sehr vielfältig einsetzen. Mit Preact existiert auch eine extrem kompakte Variante.

Mit Angular 2 orientiert sich die zweite Version des bekannten Webframeworks ebenfalls an einem Komponentenmodell. Für Fans von Angular 1, die keine Scheu davor haben, sich neu einzuarbeiten, ist Angular 2 daher eine sinnvolle Weiterentwicklung. TypeScript wird zwar empfohlen, ist aber nicht zwingend notwendig.

Innerhalb der vielen verschiedenen Optionen, mit deren Hilfe sich Webanwendungen komponentenbasiert entwickeln lassen, sollte eigentlich für jeden Webentwickler etwas Passendes dabei sein. ■

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

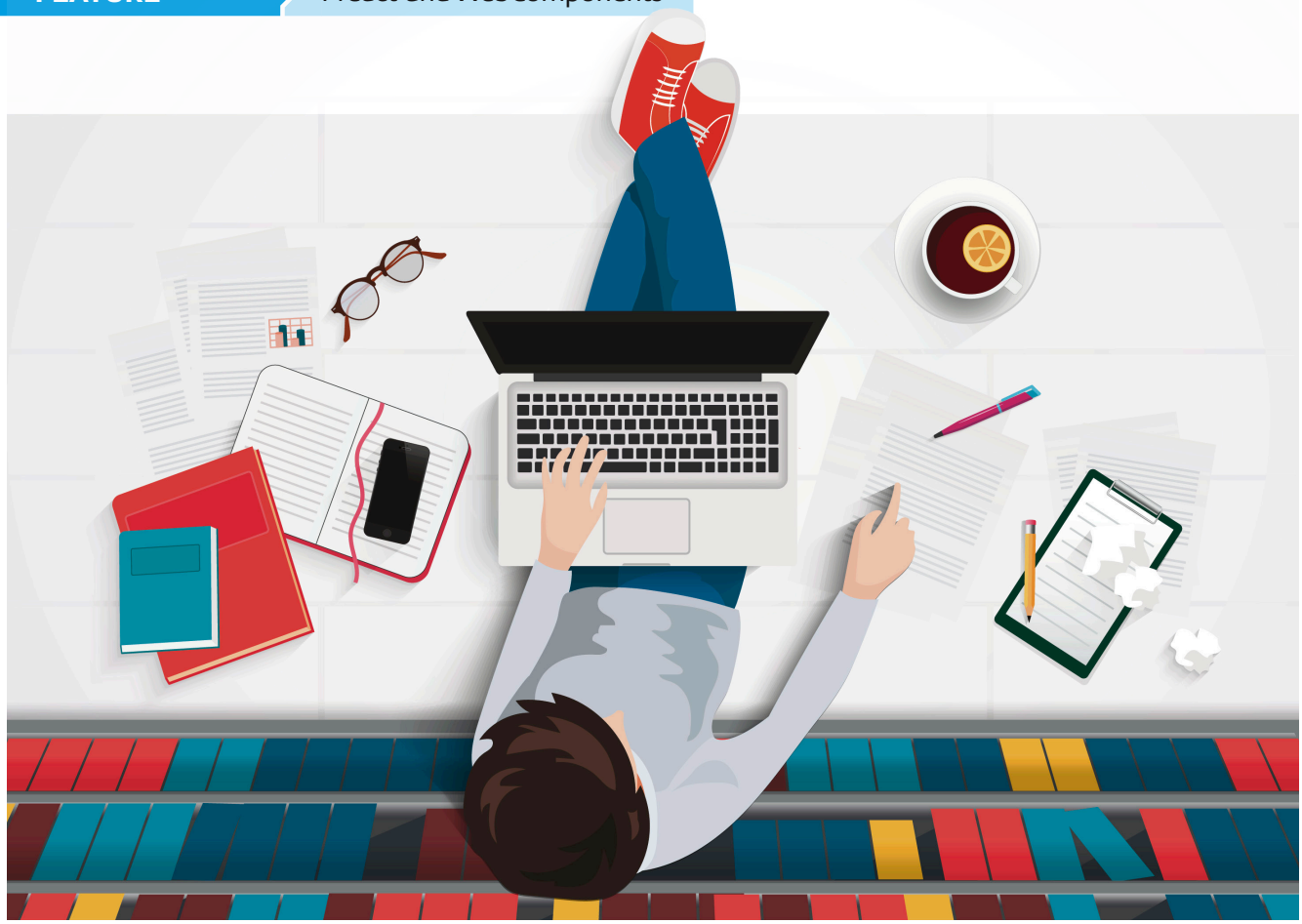


Foto: Shutterstock / Lemberg Vector Studio

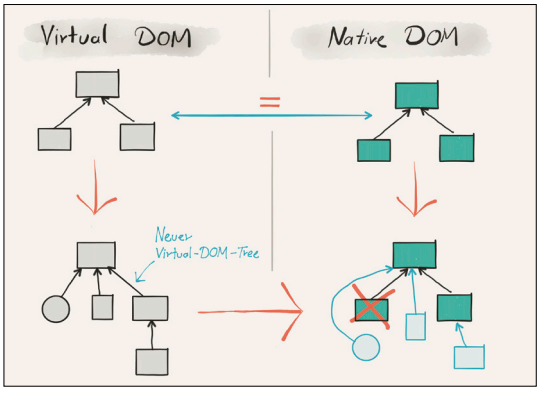
PREACT

React in drei KByte?

Ist es möglich, eine UI-Bibliothek wie React in lediglich drei KByte Code nachzubauen?

Das Projekt Preact verspricht genau das. Dieser Artikel zeigt, was an der Aussage dran ist und welche Einschränkungen sich dabei möglicherweise ergeben.

Forks und Clones sind in der Open-Source-Community oft nicht gern gesehen. Üblicherweise erwartet die Gemeinschaft eine Beteiligung am Projekt. Oft werden Bedenken geäußert, dass alternative Projekte die knappen Kräfte auf zu viele Bereiche verteilt. Berühmte Forks sind XEmacs versus GNU Emacs, Ubuntu versus Debian oder WebKit versus KHTML. Clones sind ähnlich: Auch hier entsteht ein alternatives Projekt, aber die Codebasis basiert im Gegensatz zum Fork nicht auf dem älteren Projekt. Ein Beispiel dafür ist das Open-Source-Projekt Samba, das Datei- und Druckdienste implementiert, die kompatibel zu Microsofts Domain-Controller mit Active Directory und dem SMB/CIFS-Protokoll sind.



Abgleich des Virtual DOM mit dem Native DOM (Bild 1)

Das kleine Open-Source-Projekt Preact ist ein Clone der UI-Bibliothek React von Facebook. Der Autor von Preact ist Jason Miller, ein Entwickler des US-amerikanischen Unternehmens Synacor, das Webbrowser-basierte Anwendungen und Dienste für Unternehmenskunden aus dem Umfeld Video, Internet und Kommunikationsdienste entwickelt. In einer zehn Jahre alten, auf Backbone und jQuery basierten Website mit mehreren Millionen Nutzern wollte Jason Miller dennoch die Vorteile von React und dessen komponentenbasiertem Modell nutzen. So entwickelte er mit Preact eine weitgehend zum React-API kompatible Bibliothek und konzentrierte sich dabei insbesondere auf zwei Aspekte: Sie sollte extrem klein sein und sie sollte schnell sein. Nur auf diese Weise konnte er die Bibliothek in diesem Projekt nutzen, ohne die Webanwendung unnötig aufzublähen.

Bei React handelt es sich um eine JavaScript-Bibliothek, die das Erstellen von Bedienoberflächen erleichtern soll. Der Programmierer definiert einzelne Komponenten und legt deren Verhalten fest. Die komponentenorientierte Programmierweise erleichtert die Wiederverwendung. Eine der größten Besonderheiten von React ist das sogenannte virtuelle DOM (Bild 1). Statt einer klassischen imperativen Programmierschnittstelle wie beim nativen Browser-DOM besitzt das React-vDOM ein funktionales API. Man kann es nicht modifizieren, sondern erzeugt es stets von Grund auf.

Das klingt möglicherweise teuer, ist aber durch die leichtgewichtige Implementierung tatsächlich rasend schnell. Durch einen intelligenten Vergleichsalgorithmus kann React ein solchermaßen erstelltes virtuelles DOM auf das gerade im Browser gerenderte native DOM anwenden. Dabei wird das native DOM in möglichst wenig Schritten so modifiziert, dass es die Struktur und Inhalt des vDOM widerspiegelt. Der Vorteil: Der React-Programmierer kann aus einem beliebigen Anwendungszustand einfach ein vollständiges virtuelles DOM erzeugen und dieses dennoch effizient rendern lassen. Er tut so, als ob er die Anwendung stets von Grund auf rendert, aber in Wirklichkeit ändert sich nur alles Notwendige. Dieser Ansatz führt zu wesentlich robusteren Anwendungen und vereinfacht die Entwicklung von komplexen Bedienoberflächen enorm. Jeder Zustand der Bedienoberfläche wird aus einem Anwendungszustand berechnet und ist damit jederzeit wiederherstellbar.

Facebook selbst setzt React nahezu überall ein. Schon vor zwei Jahren berichteten Entwickler des Unternehmens auf der LondonReact-Konferenz davon, dass man nunmehr 10.000 React-Komponenten in der Codebasis habe und dass sie React für alle neuen Projekte einsetzen. Selbst die Entwicklung nativer Anwendungen ist in React möglich: Mit dem Projekt React Native können Entwickler ihr React-Wissen einsetzen, um native Apps für iOS, Android und Windows 10 zu entwickeln.

Oft umstritten

Trotz seines großen Erfolgs hat React von Beginn an polarisiert. Einige der technischen Entscheidungen schienen den etablierten Regeln völlig entgegenzustehen. In den Anfängen drückten viele Entwickler ihr Unverständnis aus, als sie sahen, dass React mit der Syntaxerweiterung JSX HTML-artige Schnipsel direkt in den Quellcode einbettet.

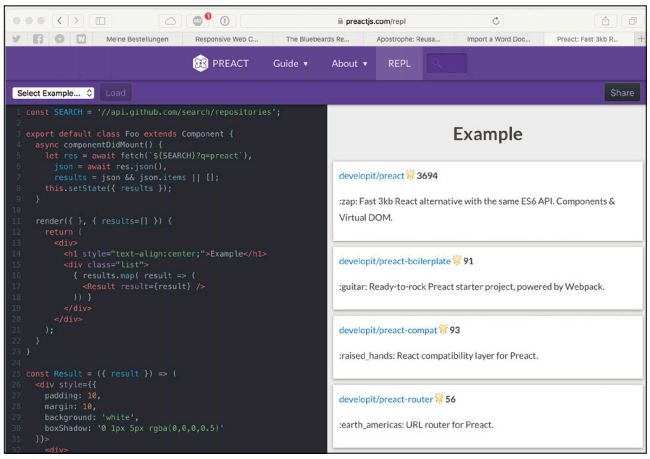
Die Befürchtung: Werden so nicht wieder Logik und Templates vermischt, wie man das eigentlich Jahre zuvor bereits abgestreift wähnte? Doch in der Praxis erwies sich der Ansatz als sinnvoll: Selten werden Templates ohne die zugehörige Logik wiederverwendet; meist gibt es tatsächlich zu jeder Komponente genau ein Template. Wenn dieses unbedingt in einer eigenen Datei untergebracht werden muss, ist das oft eher hinderlich. Außerdem stellte man bei React fest, dass diese JSX-Snippets oft äußerst klein sind. Sobald sie zu groß werden, teilt man sie wieder in eigene Komponenten auf. Zu guter Letzt ist JSX eigentlich gar kein Template-System im herkömmlichen Sinne, sondern lediglich eine syntaktische Vereinfachung von verschachtelten Funktionsaufrufen.

Auch die Verwendung von *onClick*-Eigenschaften *(props)* für Ereignis-Handler oder Inline-Styling per *style*-Eigenschaft sorgten für Diskussionen. Doch die Erfahrung bei der Entwicklung mit React zeigte auch hier, dass derartige Ansätze durchaus Vorteile bieten können.

Alternativen erscheinen

In der Open-Source-JavaScript-Welt zeigt sich eines besonders: Sobald ein neuartiger Ansatz sich als erfolgreich erweist, dauert es nicht lange, und schon erscheinen erste Varianten davon. Oft starten diese als Experiment. Gründe dafür können sein, dass Entwickler mit manchen Details einer Bibliothek wie React nicht einverstanden sind oder schlicht und einfach einer Not-invented-here-Gewohnheit folgen.

Die Liste der durch React inspirierten Bibliotheken und Frameworks ist lang: Vor etwa drei Jahren erschien mit Riot eine deutlich an React angelehnte Bibliothek. Auch Cycle.js folgte dem durch React aufgezeigten Grundprinzip und setzte es in deutlich abgewandelter Form um. Sogar Angular 2 hat als Framework teils mehr mit React gemein als mit seinem direkten Vorgänger. Ob React dabei wirklich der erste Vertreter dieser Art war, kann man durchaus bezweifeln, aber es ist bis heute die erfolgreichste und am meisten verbreitete Implementierung.



Preact-Website mit Live-REPL (Bild 2)

Was ist Preact?

Die Website zu Preact (Bild 2) stellt die Bibliothek als Versuch vor, die Kernvorteile von React mit so wenig Code wie möglich nachzubauen, damit es in bestehenden Anwendungen oder bei sehr knappen Ressourcen ohne Probleme integriert werden kann. Die (kleine) Größe der Bibliothek ist ihr Alleinstellungsmerkmal und damit auch so etwas wie eine Daseinsberechtigung neben dem offiziellen React. Minifiziert und mit GZip gepackt soll die Bibliothek gerade einmal 3 KByte groß sein.

Die Idee war nicht völlig neu: Auch Riot.js wurde damit beworben, eine sehr kleine, von React inspirierte Bibliothek zu sein. Mit circa 9 KByte wäre sie jedoch immer noch drei- ▶

mal so groß wie Preact. Außerdem weicht das API von Riot.js sehr stark von React ab; Preact hingegen ist in weiten Teilen kompatibel zum offiziellen React-API. Zusammen mit der Größe ist also die Kompatibilität zum offiziellen React das zweite Merkmal, das für Preact spricht. Die beste Möglichkeit, Preact mit dem Original zu vergleichen, besteht in einem minimalen Code-Beispiel, das mit beiden Bibliotheken realisiert wird und zeigt, wie ähnlich sie sich im Idealfall verhalten.

Das folgende kleine HalloWelt-Beispiel benötigt mit Webpack gebundelt und minifiziert 145 KByte:

```
// Listing src/index.js
// Einstiegspunkt der App

import React from 'react'
import {render} from 'react-dom'
import Hello from './hello'
render(<Hello name="World" />, document.body)
```

Der Einstiegspunkt der App rendert eine einfache React-Komponente *Hello* in den Body des HTML-Dokuments. Das Listing *src/hello.js* implementiert diese Komponente:

```
// Listing src/hello.js

import React from 'react'
import {render} from 'react-dom'

export default function Hello (props) {
  return <h1>Hello {props.name}</h1>
}
```

Die *webpack.config.js* wurde zum Bauen genutzt (Listing 1). Ein für React essenzieller Schritt ist dabei folgender Teil:

```
  plugins: [
    new webpack.DefinePlugin({
      'process.env.NODE_ENV':
      JSON.stringify('production')
    }),
```

Damit wird sichergestellt, dass React im Produktivmodus in das Bundle integriert wird. Das macht nicht nur das resultierende Bundle signifikant kleiner, React ist dann auch wesentlich schneller.

Listing 1: webpack.config.js

```
'use strict'

let webpack = require('webpack')
let path = require('path')
let HtmlWebpackPlugin = require('html-webpack-plugin')

let srcPath = path.join(__dirname, "src");

module.exports = {
  bail: true,
  devtool: "source-map",
  context: srcPath,
  entry: ['./index'],
  output: {
    path: path.join(__dirname, 'dist'),
    filename: "bundle.js"
  },
  module: {
    loaders: [
      {
        test: /\.js$/,
        include: srcPath,
        loader: 'babel'
      }
    ]
  },
  plugins: [
    new webpack.DefinePlugin({
      'process.env.NODE_ENV':
      JSON.stringify('production')
    }),
    new HtmlWebpackPlugin({
      template: './index.html',
      minify: { collapseWhitespace: true }
    }),
    new webpack.optimize.OccurrenceOrderPlugin(),
    new webpack.optimize.DedupePlugin(),
    new webpack.optimize.UglifyJsPlugin({
      compress: {
        screw_ie8: true,
        warnings: false
      },
      mangle: {
        screw_ie8: true
      },
      output: {
        comments: false,
        screw_ie8: true
      }
    })
  ],
  node: {
    global: true,
    process: false,
    Buffer: false,
    __filename: false,
    __dirname: false,
    setImmediate: false
  },
};
```

Komprimiert man das Bundle noch mit GZip, so schrumpft es von 145 KByte auf 41 KByte zusammen. Vergisst man allerdings den Produktivmodus, so erhält man langsameren Code, da sämtliche Laufzeitchecks des Entwicklungsmodus noch vorhanden sind. Aber auch die Bundlegröße steigt auf 202 KByte und lässt sich per GZip auf 59 KByte schrumpfen. Das sind immerhin etwa 30 Prozent mehr Daten pro Abruf des Bundles. Preact verspricht dagegen 3 KByte in der mit GZip komprimierten Form. Die Migration des Beispiels wird zeigen, ob das stimmt.

Migration von React nach Preact

Es gibt mehrere Möglichkeiten, wie man ein React-Projekt nach Preact migrieren kann. Die naheliegende wäre wohl, die Anwendung einfach umzuschreiben. Anstelle der Module *react* und *react-dom* bindet man das Modul *preact* ein. Das *HelloWorld*-Beispiel sähe dann so aus:

```
// Listing src/index.js
// Einstiegspunkt der App

/** @jsx h */
import {h, render} from 'preact'
import Hello from './hello'
render(<Hello name="World" />, document.body)

// Listing src/hello.js

/** @jsx h */
import {h} from 'preact'
export default function Hello (props) {
  return <h1>Hello {props.name}</h1>
}
```

Die Import-Anweisungen mussten geändert werden. Die Funktion *render* befindet sich nicht wie bei React in einem eigenständigen Modul *(ReactDOM)*. Anstelle des Bezeichners *React* wird außerdem *h* importiert. Das liegt daran, dass JSX-Code für Preact anders generiert werden muss. Das JSX-Schnipsel

```
<h1>Hello {props.name}</h1>
```

wird von Babel für React in folgenden Code übersetzt:

```
React.createElement(
  "h1", null, "Hello ", props.name
);
```

Preact hingegen erwartet folgenden Code:

```
h(
  "1", null, "Hello ", props.name
);
```

Die Struktur der Aufrufe ist identisch, aber statt *React.createElement* wird der Name *h* verwendet, um Elemente zu erstellen. Damit Babel den JSX-Code mit *h* übersetzt, kann man wie in den Listings gezeigt einen Pragma-Kommentar dieser Form angeben:

```
/** @jsx h */
```

Dieser weist Babel an, beim Übersetzen von JSX anstelle des Standardwerts *React.createElement* den angegebenen Wert *h* auszugeben.

Sowohl bei React als auch bei Preact ist wichtig, dass die jeweils für JSX verwendete Funktion per Import eingebunden wird. Deshalb muss bei *React.createElement* die Variable *React* definiert sein, und entsprechend ebenso bei *h*.

Das resultierende Bundle mit *preact* statt *react* und *react-dom* ist lediglich 10,2 KByte groß. Mit GZip komprimiert schrumpft es auf 4,1 KByte zusammen. Die dem Preact-NPM-Modul beiliegende minifizierte Preact-Library ist mit GZip komprimiert 3,9 KByte groß. Das entspricht zwar nicht ganz den auf der Website versprochenen 3 KByte, aber es ist dennoch ein beeindruckender Wert.

Zu welchem Preis?

Bei einem Größenunterschied mit einem Faktor von fast exakt 10 zwischen React und Preact muss es eigentlich signifikante Unterschiede geben. Ein erster Unterschied wird auf der Website genannt: Preact implementiert ausschließlich Stateless-Functional-Components und ES6-Klassen. Das ursprüngliche Komponenten-API mit *React.createClass()* wird nicht unterstützt.

Das *HalloWelt*-Beispiel verwendete lediglich eine Stateless-Functional-Component – also eine einfache Funktion, die als Parameter die Props erhält und als Ergebnis das generierte vDOM liefert. Würde man dieselbe Komponente mit *React.createClass()* implementieren, sähe das so aus:

```
import React, {createClass} from 'react'
import {render} from 'react-dom'

let Hello = createClass({
  render: function () {
    return <h1>Hello {this.props.name}</h1>
  }
})

export default Hello
```

Ersetzt man auch hier wieder die Module *react* und *react-dom* durch *preact,* so kommt es zu einem Fehler:

```
bundle.js:1 Uncaught TypeError: (0 , r.createClass) is
not a function
```

Es gibt kein Äquivalent zu *createClass* in Preact. Genau das ist gemeint, wenn es heißt, dass Preact nur Stateless-Functional-Components und ES2015-Klassen unterstützt.

Ein weiterer Unterschied ist das React.Children-API: In React sind die Kinder einer Komponente entweder ein Array ▶

aus Komponenten oder eine einzelne Komponente. Damit soll für den sehr häufigen Fall einzelner Kinder jeweils ein unnötiges Array-Objekt gespart werden. Allerdings hat das zur Folge, dass man vor einem Zugriff auf *props.children* stets prüfen muss, ob es sich um eine einzelne Komponente oder ein Array handelt.

React bietet deshalb das sogenannte React.Children-API mit den Funktionen *React.Children.map()*, *React.Children.forEach()*, *React.Children.count()*, *React.Children.only()* und *React.Children.toArray()* an. Preact handhabt dies wesentlich simpler: Die Kinder einer Komponente werden einfach stets als Array repräsentiert. Das macht jedoch React-Code, der das React.Children-API nutzt, wiederum nicht kompatibel zu Preact.

Auch die sogenannten PropTypes fehlen in Preact: Damit kann man in React die Typen der Props einer Komponente definieren. Preact unterstützt dieses API nicht, zumindest nicht ohne Erweiterungen.

Eine der sehr bequemen Eigenschaften von React ist das Event-System. React verwendet Wrapper in Form von SyntheticEvent-Objekten. Diese vereinheitlichen die Zugriffsschnittstellen auf die nativen Ereignisobjekte. Immerhin unterstützt React nicht nur die jeweils neuesten Webbrowser, sondern bewusst auch ältere Versionen. Im Januar 2016 wurde mit React v15 der offizielle Support für IE 8 eingestellt – wer also auf Nummer sicher gehen will, sollte für IE 8 bei der React-Version v0.14 bleiben, auch wenn der Code für ältere Browserversionen in den aktuellen React-Versionen nach wie vor vorhanden ist. Diese Bestandteile des Codes tragen dazu bei, dass React signifikant größer als Preact ist.

Preact verwendet keine SyntheticEvent-Wrapper – als Entwickler ist man demnach selbst dafür verantwortlich, portablen Code in Ereignis-Handlern zu schreiben. Dennoch unterstützt auch Preact alle modernen Browser und IE ab IE 9. Sogar IE 7 und IE 8 sollen funktionieren, wenn man einige notwendige Polyfills einbindet.

Preact als Drop-in-Ersatz

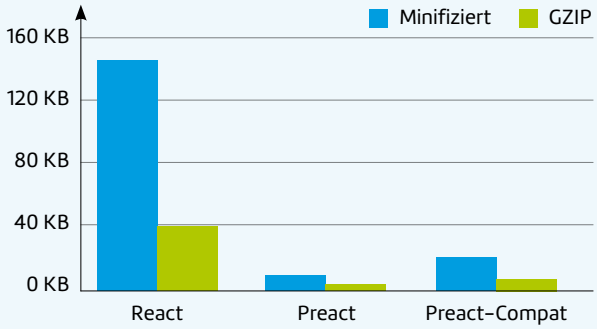
Wenn man sich das Umschreiben des eigenen React-Codes sparen oder dies zumindest vereinfachen möchte, dann hilft dabei das Modul *preact-compat*. Dabei handelt es sich um eine Kompatibilitätsschicht, die einige der fehlenden APIs aus React verfügbar macht. Dazu gehört das bereits erwähnte *React.createClass*, aber auch die PropTypes und das React.Children-API – nicht jedoch die synthetischen Events.

Das Paket *preact-compat* macht Preact damit zu einem Drop-in-Ersatz. Dazu trägt man in der *webpack.config.js* folgende Alias-Konfiguration ein:

```
resolve: {
  alias: {
    'react': 'preact-compat',
    'react-dom': 'preact-compat'
  } },
```

Webpack lädt dann sowohl statt *react* als auch statt *react-dom* jeweils das Kompatibilitätspaket *preact-compat*. Der ursprüngliche React-Code läuft dann ohne jede Codeänderung – selbst in der Fassung mit *React.createClass*. Statt 145 KByte wie bei React hat das resultierende Bundle dann 21,7 KByte. Mit GZip komprimiert bleiben noch 7,7 KByte. Bild 3 zeigt die verschiedenen Varianten im Größenverhältnis.

Volumenunterschiede



Größenvergleich: Das Diagramm zeigt die Volumenunterschiede von React- und Preact-Bundles (Bild 3)

web & mobile developer 11/2016 Quelle: Jochen Schmidt

Custom-Elemente in Preact

Komponenten werden in Preact genauso definiert wie in React, wobei die Basis-Bibliothek nur die ES2015-Klassen und die sogenannten Stateless-Function-Components realisiert. Beschränkt man sich auf diese, oder bindet man die Kompatibilitätsschicht *preact-compat* ein, dann kann man implementierte Komponenten sowohl in React als auch in Preact verwenden.

Einen Nachteil hat Preact mit React gemeinsam: Mit Komponenten kann man innerhalb des virtuellen DOM so etwas wie Custom-Elemente definieren, wie man sie im nativen DOM auch im Rahmen der WebComponents-Spezifikation kennt. Man kann sich also eigene HTML-Elemente definieren. Bei React und Preact sind diese aber auf das virtuelle DOM begrenzt und deshalb nicht im nativen DOM direkt als Element benutzbar. Wenn man also – beispielsweise in einem CMS – den Nutzer eigene HTML-Inhalte eingeben lässt, dann können die mit React oder Preact definierten Komponenten dort nicht einfach im Markup benutzt werden.

Eine Möglichkeit besteht darin, mit dem WebComponents-API jede React-Komponente zu wrappen. Das ist jedoch einerseits aufwendig, und andererseits ist der Browsersupport für WebComponents nach wie vor schlecht.

Für diesen Anwendungsfall hat der Autor von Preact eine interessante Lösung geschaffen: Mit *preact-markup* kann man innerhalb einer JavaScript-Anwendung (also im Webbrowser) HTML (oder XHTML) parsen und daraus ein virtuelles DOM für Preact erzeugen. Ein solches virtuelles DOM kann natürlich jederzeit mit dem nativen DOM synchronisiert werden.

Doch damit nicht genug: Man kann dem *preact-markup*-Parser auch eine Abbildungsvorschrift mitgeben, anhand derer er beim Parsen Namen von HTML-Elementen auf React-

Komponenten abbildet. Das ist nichts anderes als ein browserunabhängiger Mechanismus für Custom-Elemente. Der folgende Code zeigt einen LiveEditor, bei dem die Eingabe in ein Textfeld bei jedem Tastendruck dazu führt, dass der Inhalt als HTML geparst und daraus ein Virtual DOM erzeugt und darunter sofort gerendert wird. Dazu werden zwei Komponenten *Headline* und *PanicButton* definiert und mit den Tagnamen *x-headline* und *x-panic* verknüpft. Bild 4 zeigt das Ergebnis:

```
// Listing src/index.js

import {h, render} from 'preact'
import LiveEditor from './LiveEditor'
render(<LiveEditor />, document.body)
```

Die *index.js* rendert einfach nur die unten definierte LiveEditor-Komponente.

```
// Listing src/Headline.js
/** @jsx h */

import {h} from 'preact'
export default function Headline (props) {
  return <h1>Headline: {props.children} </h1>
}
```

Das Listing *Headline.js* implementiert eine sehr einfache Preact-Komponente. Der Quellcode ist ohne Modifikation nicht in einer herkömmlichen React-Anwendung einsetzbar, da er das Modul *preact* nutzt und der JSX-Code für die Funktion *h* übersetzt wird.

```
// Listing src/PanicButton.js
import React from 'preact'
```

Beispiel: Custom-Elemente werden live aus Markup im Textfeld erzeugt (Bild 4)

```
function PanicButton (props) {
  let handleClick = (ev) => {
    alert("Panic Panic Panic!")
    if (props.onClick) props.onClick(ev)
  }
  return (<button
    onClick={handleClick}>Panic
    {props.children}!
  </button>)
}
```

Im Listing *PanicButton.js* befindet sich eine React-Komponente, die eine Schaltfläche implementiert, bei deren Betätigung ein *alert()*-Dialog geöffnet wird. Der Quellcode wäre in dieser Form direkt in React einsetzbar. Durch das Aliasing auf *preact-compat* kann die Komponente jedoch ebenso in Preact eingesetzt werden.

Der JSX-Code wird hierbei in *React.createElement*-Aufrufe übersetzt. Dennoch sind sowohl *Headline* als auch *PanicButton* ohne Probleme zusammen benutzbar, denn am Ende wird aus beiden eine kompatible Virtual-DOM-Repräsentation erzeugt (Listing 2). ▶

Listing 2: LiveEditor

```
import {h, render, Component} from 'preact'

import Markup from 'preact-markup'
import Headline from './Headline'
import PanicButton from './PanicButton'

let customElements = {
  "x-headline" : Headline,
  "x-panic" : PanicButton
}

export default class LiveEditor extends Component {
  render () {
    return (
      <div className="LiveEditor">
        <h2>Editor</h2>
        <textarea
          value={this.state.markup}
          onKeyUp=
          {this.handleChange.bind(this)}/>
        <h2>Live Preview</h2>
        <Markup type="html"
        markup={this.state.markup}
        components={customElements}/>
      </div>
    )
  }

  handleChange () {
    this.setState({
      markup: event.target.value
    });
  }
}
```

Die Komponente *LiveEditor* nutzt die Komponente *Markup* aus *preact-markup,* um an dieser Stelle einen über die Prop *markup* übergebene HTML-Zeichenkette zu parsen und den resultierenden Virtual-DOM-Baum zu rendern. Jedes Mal, wenn man in der *textarea* etwas tippt *(keyUp*-Ereignis), wird der Inhalt des Textfelds als Eigenschaft im State des LiveEditors gesetzt. Durch *setState()* wird automatisch ein Rendern der Komponente *LiveEditor* getriggert, und *Markup* erhält beim neuen Render-Vorgang seinen HTML-String aus dem gerade aktuellen State. Werden im eingegebenen HTML die Elementnamen *x-headline* oder *x-panic* verwendet, so werden automatisch Headline- oder PanicButton-Komponenten erzeugt.

Der HTML-Code, der in die Markup-Komponente gesteckt wird, kann natürlich auch das Ergebnis eines Ajax-Requests sein, der den Inhalt aus einer Datenbank liest. Die Quelle ist eigentlich egal. Auf diese Weise kann man auch Inhalte dynamisch ausgeben, die in serverseitigen Templates generiert werden, und dort die durch React-Komponenten implementierten Custom-Elemente nutzen.

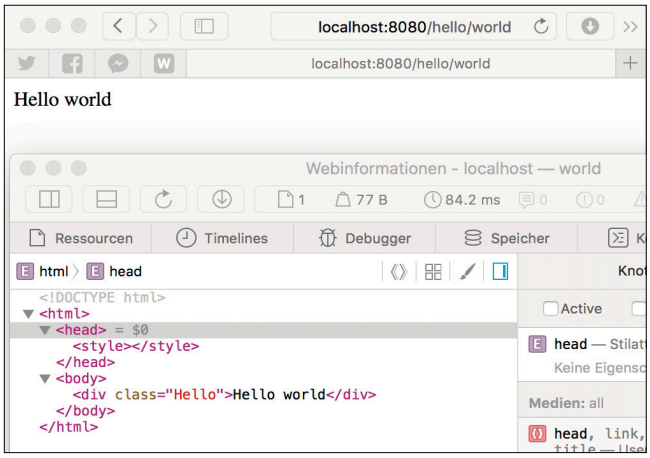
Server-side Preact

React ist dafür bekannt, dass es auch serverseitig eingesetzt werden kann. Dasselbe gilt auch für Preact. Der Grund dafür ist sehr einfach: Da Preact ebenfalls ein virtuelles DOM verwendet, das unabhängig vom nativen Browser-DOM in JavaScript implementiert ist, kann der vDOM-Code auch in Node.js auf dem Server laufen. Die normale *render*-Funktion vergleicht das virtuelle DOM mit dem nativen DOM und modifiziert Letzteres exakt so, dass es die Struktur und Inhalte des virtuellen DOM aufweist. Man kann jedoch diese virtuelle DOM-Struktur auch auf ganz andere Weise rendern:

Das Modul *preact-render-to-string* stellt einen Renderer bereit, der ein Preact-Virtual-DOM als Zeichenkette übersetzt. Das Ergebnis kann man natürlich auch in einem *express()*-Endpunkt ausliefern:

```
'use strict'
import express from 'express'
import {h} from 'preact'
import render from 'preact-render-to-string'
/** @jsx h */
const Hello = (props) => (
  <div class="Hello">Hello {props.name}</div>
)
let app = express()
app.listen(8080)

app.get('/hello/:name', (req,res) => {
  console.log("Hello "+req.params.name)
  let html = render(<Hello name={req.params.name} />)
  res.send('<!DOCTYPE html><html><body>${html}</body>
  </html>')
})
```

Bild 5 zeigt das Ergebnis der Anfrage des *express*-Endpunkts im Webbrowser und den Entwickler-Tools.



Serverseitiges Preact mit Express als HTTP-Server (Bild 5)

Auf diese Weise kann man React-Komponenten auch serverseitig nutzen – ähnlich einer serverseitigen Template-Engine. Die Krönung des Ganzen ist erreicht, wenn das initiale Rendering einer Webanwendung serverseitig erfolgt und somit bereits fertig am Webbrowser ankommt, jedoch weitere *render()*-Vorgänge im Webbrowser durchgeführt werden können. Man nennt dies dann universelle Webanwendung.

Fazit

Preact ist ein beeindruckendes Beispiel dafür, wie man mit extrem wenig Quellcode sehr viel erreichen kann. Im Resultat entsteht eine Bibliothek, die man tatsächlich mit gutem Gewissen als Abhängigkeit einbinden kann.

Ein vollständiger Ersatz für das offizielle React ist Preact jedoch nicht zwingend. React enthält wesentlich mehr Code, um die Cross-Plattform-Unterschiede der verschiedenen Browser auszugleichen. Auch was hilfreiche Meldungen für Entwickler und die Integration mit einer Vielzahl von Entwicklerwerkzeugen angeht, kann React gegenüber seinem kleinen Bruder deutlich punkten.

Eine Idee wäre auch, noch zur Entwicklungszeit React einzusetzen und dann einen produktiven Build mit Preact durchführen – doch nicht immer ist das wirklich praktikabel. Außerdem ist React selbst in minifizierter und mit Gzip komprimierter Form nicht ungewöhnlich groß. Für einen Entwickler, der eine React-Anwendung schreibt, ist das offizielle React-Projekt also in der Regel die bessere Wahl. Man spart zwar ein paar Bytes weniger, dafür aber mitunter deutlich Zeit bei der Entwicklung. ■

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

developer media

developer media Trainings

Updates für Ihr Know-How

„Das Entwickeln von qualitativ hochwertiger Software bedeutet auch, sich selbst ständig weiterzuentwickeln und nicht still zu stehen."

David Tielke
Softwareentwickler, Berater, Trainer

Codequalität mit JavaScript

Trainer: Golo Roden

3 Tage, 02.-04.11.2016, Köln

Ab 2.199 EUR zzgl. MwSt.

Moderne Webentwicklung mit ASP.NET Core

Trainer: David Tielke

3 Tage, 21.-23.11.2016, Köln

Ab 2.199 EUR zzgl. MwSt.

PHPUnit erfolgreich einsetzen

Trainer: Sebastian Bergmann

2 Tage, 07.-08.11.2016, Köln

Ab 1.799 EUR zzgl. MwSt.

Angular 2 mit TypeScript 2

Trainer: Johannes Hoppe

3 Tage, 21.-23.11.2016, Köln

Ab 2.199 EUR zzgl. MwSt.

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

developer-media.de/trainings

Bild: shutterstock / garagestock

ZEICHNEN IN HTML5

Zwei Technologien

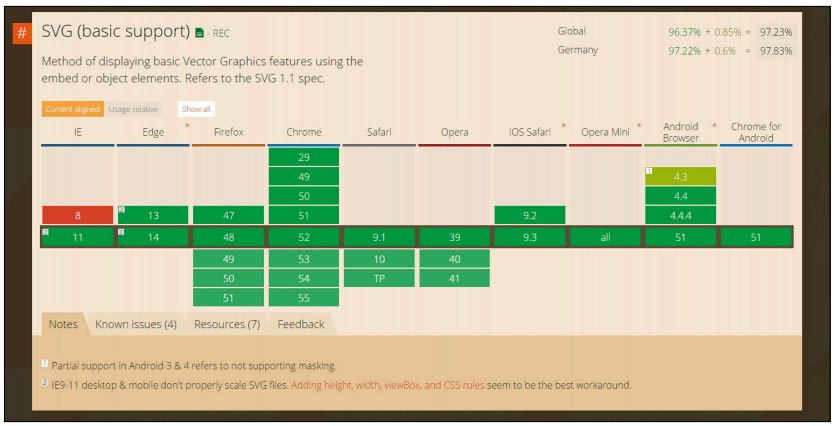
Zum Zeichen in HTML5 stehen Entwicklern zwei Technologien zur Verfügung: SVG mit dem HTML-Tag <svg> und Canvas unter Verwendung des HTML-Tags <Canvas>.

Dieser Artikel behandelt die grundlegenden Formen und Stilmittel der HTML-Elemente Canvas und SVG. Hierbei werden die beiden Techniken einander gegenübergestellt und Beispiele gezeigt, wie man als Entwickler mit den beiden Instrumenten ein identisches Ergebnis auf einer Webseite erzielen kann. Canvas wurde für das Mac-OS-Dashboard entwickelt und später auch im Safari-Browser zur Verfügung gestellt. Heute ist es Bestandteil der HTML5-Spezifikation. Canvas wird mit dem HTML-Tag *<canvas>* implementiert:

```
<canvas height="600" width="400"
id="canvas"> Browser unterstützt das
Element nicht</canvas>
```

Mit Hilfe von JavaScript können Entwickler den Canvas-Kontext laden und anschließend damit Grafiken zeichnen.

```
var canvas = document.getElementById('canvas');
```



Browsercheck auf SVG-Unterstützung (Bild 1)

```
// Holt den 2D-Context in die Variable ctx
// Es wird auf dem Kontext und nicht auf
// dem <canvas>-Element gezeichnet

var ctx = canvas.getContext("2d");
if (ctx) {
  // Zeichenlogik.
}
```

Browsercheck auf Canvas-Unterstützung (Bild 2)

Der Canvas-Kontext stellt Funktionen zum Zeichnen von Linien, Rechtecken, Kreisen, Bögen, Texten und Bildern bereit. Mit Canvas sind Entwickler nicht nur auf das Zeichnen von einfachen Grafiken beschränkt, sondern sie können darüber hinaus auch Bild- und Videomanipulationen vornehmen und Verläufe darstellen. Bei Canvas werden Grafiken direkt geändert und nicht gespeichert, das heißt, ein Entwickler muss erforderliche Funktionen erneut aufrufen, um zum Beispiel beim neuen Skalieren des gezeichneten Kontexts die Qualität des Gesamtergebnisses beizubehalten. Dies entspricht in etwa dem Vorgehen von gängigen Grafikprogrammen wie zum Beispiel Photoshop.

Grundlegende Anmerkungen zu SVG

SVG bietet Entwicklern ein Grafikmodell mit Speichermodus. SVG ist Teil des HTML-DOM und vollständig in HTML integriert. Somit ist sowohl mit JavaScript und DOM-Manipulation als auch mit HMTL-Markup das Erreichen ein und desselben Ergebnisses möglich. Für die Nutzung von SVG wird das HTML-Tag *<svg>* verwendet:

```
<svg height="410" width="800" id="svg">
  <!-- Markup-->
</svg>
```

Mit SVG und mit Canvas können die gleichen Ergebnisse erzielt werden.

Implementierung

Die folgenden Abschnitte zeigen in Beispielen, wie sich durch Anwendung beider Techniken dasselbe Ergebnis erzielen lässt. Vor der Wahl der Technik sollte der Browsersupport für SVG (Bild 1) und Canvas (Bild 2) überprüft werden.

In den meisten Canvas-Beispielen finden die JavaScript-Befehle *beginPath, moveTo* und *stroke* des Canvas-Kontexts Verwendung: *beginPath* löscht alle bisher verwendeten Pfadinformationen im Kontext, und *moveTo* setzt einen neuen Referenzpunkt auf dem Kontext, ohne eine Linie zu ziehen, das heißt, es ist vergleichbar mit dem Absetzen eines Stifts und seinem erneuten Ansetzen an einer anderen Stelle. *stroke* verbindet alle bisher erstellten Pfade mit einer Linie.

Zu jeder Canvas-Implementierung gibt es die Darstellung der alternativen Anwendung von SVG. Die SVG-Beispiele bestehen in den meisten Fällen aus HTML-Markups, auf die im Folgenden in den einzelnen Beispielen detailliert eingegangen wird.

Die Verwendung beider Techniken fordert die Beachtung der Positionierung. Der Entwickler muss beachten, dass es sich um ein umgekehrtes Koordinatensystem handelt. Die linke obere Ecke hat die Koordinaten *0,0* und die rechte untere Ecke besitzt in einem Canvas von 200 Pixel Breite und 200 Pixel Höhe, wie in Bild 3 dargestellt, die Koordinaten *200/200.*

Gestaltung eines Textes

Um mit Canvas einen Text darzustellen, gibt es zwei verschiedene Funktionen: *strokeText (text, x, y[, maxWidth])* zeichnet die Umrisse eines Textes mit der in *strokeStyle* hinterlegten Textfarbe sowie der in *font* hinterlegten Schriftart und ▶

Listing 1: Canvas-Text

```
<canvas id="canvas" width="600" height="400">
</canvas>
<script>
  var canvas = document.getElementById('canvas');
  var ctx = canvas.getContext("2d");
  if (ctx) {

    // Löschen aller bisher im Path-Objekt
    // gespeicherten Punkte
    ctx.beginPath();

    // Als String gespeicherte CSS-konforme Font-
    // Angabe
    ctx.font = "30px arial";

    // Setzen der Füllfarbe
    ctx.fillStyle = "red";

    // fillText(text, x, y, maxWidth): Füllfarbe
    // wird die in fillStyle hinterlegte Füllfarbe
    // verwendet.
    ctx.fillText("Nice job Canvas", 0, 390);
  }
</script>
```

Schriftgröße. *fillText (text, x, y[, maxWidth])* zeichnet einen gefüllten Text mit der in *fillStyle* hinterlegten Textfarbe sowie der in *font* hinterlegten Schriftart und Schriftgröße.

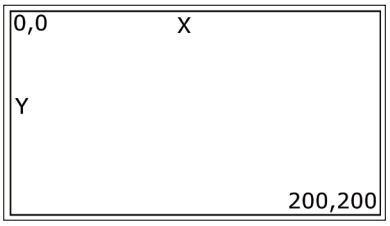
Um den Text zu formatieren, können im Canvas-Kontext die Attribute *font* und bei Verwendung der Funktion *strokeText*, *strokeStyle* oder *fillStyle* bei dem Aufruf der Funktion *fillText* genutzt werden. Listing 1 zeigt eine Implementation, in der ein roter gefüllter Text mit der Schriftgröße 30 Pixel in der Schriftart Arial gezeichnet wird (Bild 4).

In SVG wird unter Verwendung des HTML-Tags *<text>* ein gleiches beziehungsweise ähnliches Ergebnis (Bild 5) erzielt. Mit den Attributen x, y erzielt man die genaue Positionierung und über CSS-Eigenschaften erreicht man eine Formatierung:

```
<svg height="30" width="200">
  <text x="0" y="15" fill="red"
  style="font-family:
  Arial; font-size: 30px">Nice job SVG</text>
</svg>
```

Listing 2: Canvas-Linien

```
<canvas id="canvas" width="600" height="400">
</canvas>
<script>
  var canvas = document.getElementById('canvas');
  var ctx = canvas.getContext("2d");
  if (ctx) {
    // Löschen aller Punkte im Path-Objekt
    ctx.beginPath();
    // Setzen des Referenzpunkts, ohne dabei eine
    // Linie zu zeichnen. (x|y)
    ctx.moveTo(10, 10);

    // Zeichnen einer Linie vom aktuellen
    // Referenzpunkt zum neuen (x|y)
    ctx.lineTo(225, 350);
    // Zeichnen einer Linie vom aktuellen
    // Referenzpunkt zum neuen (x|y)
    ctx.lineTo(300, 10);
    // Zeichnen einer Linie vom aktuellen
    // Referenzpunkt zum neuen (x|y)
    ctx.lineTo(400, 350);

    // Setzen der Breite der zu zeichnenden Linie
    ctx.lineWidth = 5;
    // Setzen der Formatvorlage
    ctx.strokeStyle = 'black';
    // Zeichnen aller aktuellen Pfade
    ctx.stroke();
  }
</script>
```



Die Verwendung beider Techniken fordert es, die Positionierung zu beachten (Bild 3)

Text darstellen mit Hilfe von Canvas (Bild 4)

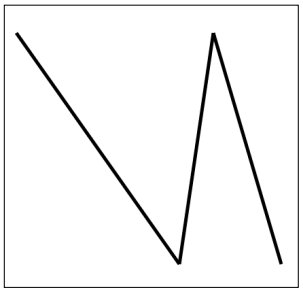
Text darstellen mit SVG (Bild 5)

Linien mit Canvas und SVG

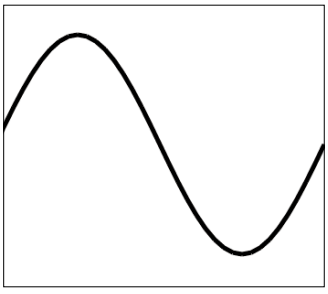
Zum Zeichnen von einfachen Linien stellt der Canvas-Kontext Entwicklern die Funktion *lineTo(x,y)* bereit. Die Funktion beschreibt einen Pfad vom letzten Referenzpunkt zu den in *lineTo* definierten Koordinaten, die den neuen Referenzpunkt festlegen. Mit dem Aufruf der Funktion *stroke* werden diese Referenzpunkte verbunden. Listing 2 produziert das gewünschte Ergebnis (Bild 6).

Der im Canvas produzierte Output wird in SVG mit dem HTML-Tag *<line>* erreicht. Die Attribute *x1*, *y1* definieren den Anfangspunkt. Den Endpunkt beschreiben *x2*, *y2*. Im folgenden Listening wird dies veranschaulicht – es werden drei aneinandergrenzende Linien gezeichnet:

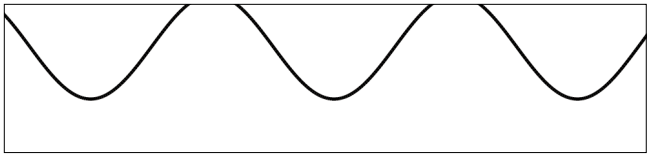
```
<svg height="400" width="500">
  <!-- schwarze Linien -->
  <line x1="10" y1="10" x2="250" y2="350"
  style="stroke: black; stroke-width: 5px;" />
  <line x1="250" y1="350" x2="300" y2="10"
  style="stroke: black; stroke-width: 5px;" />
  <line x1="300" y1="10" x2="400" y2="350"
  style="stroke: black; stroke-width: 5px;" />
</svg>
```



Linie: Eine einfache Linie mit Canvas zeichnen (Bild 6)



Eine Sinuskurve mit Hilfe von Canvas zeichnen (Bild 7)



Eine Sinuskurve mit Hilfe von SVG und JavaScript erstellen (Bild 8)

Um beispielsweise eine Sinuskurve in Canvas zu zeichnen, kann ein Entwickler die Kurve Punkt für Punkt, wie in Listing 3 dargestellt, mit der Funktion *lineTo* zeichnen (Bild 7).

SVG erlaubt Entwicklern auch, über JavaScript Elemente hinzuzufügen. Dies geschieht durch DOM-Manipulation. In Listing 4 wird das SVG-Element über *getElementById* in einer Variablen *svg* gespeichert. Innerhalb einer *for*-Schleife wird bei jeder Iteration mittels *createElementNS* ein neues <*line*>-Element erstellt und nach den Berechnungen der Sinuskurve mit *appendChild* dem existierenden <*svg*>-Element hinzugefügt. Dies erzeugt den in Bild 8 dargestellten Output.

Zeichnen von Verläufen

Eine lineare Verlaufsdarstellung, umgesetzt in Canvas, wird in Listing 5 verdeutlicht. Über die Funktion *rect* wird ein Rechteck von 250 Pixel Breite und 175 Pixel Höhe erstellt. Anschließend wird ein lineares Verlaufs-Objekt mit der Kontextfunktion *createLinearGradient (x0,y0,x1,y1)* erstellt. *x0* und *y0* definieren die Startposition des Verlaufs und *x1* und *y1* den Endpunkt. Mit der Funktion *addColorStop (offset, color)* ▶

Listing 3: Sinuskurve in Canvas

```
<canvas id="canvas" width="600" height="400">
</canvas>
<script>
  var canvas = document.getElementById('canvas');
  var ctx = canvas.getContext("2d");
  if (ctx) {
    // Löschen aller bisher im Path-Objekt
    // gespeicherten Punkte
    ctx.beginPath();
    // Setzen des Referenzpunkts, ohne dabei eine
    // Linie zu zeichnen. (x|y)
    ctx.moveTo(10, 10);
    var i;
    var counter = 0, x = 0, y = 180;
    // 360 Iterationen
    var increase = 90 / 180 * Math.PI / 9;
    for (i = 0; i <= 360; i += 10) {
      ctx.moveTo(x, y);
      x = i;
      y = 180 - Math.sin(counter) * 120;
      counter += increase;
      ctx.lineTo(x, y);
      ctx.stroke();
    }
    // Breite der zu zeichnenden Linie
    ctx.lineWidth = 5;
    // Setzen der Formatvorlage
    ctx.strokeStyle = '#f00';
    // Zeichnen aller aktuellen Pfade
    ctx.stroke();
  }
</script>
```

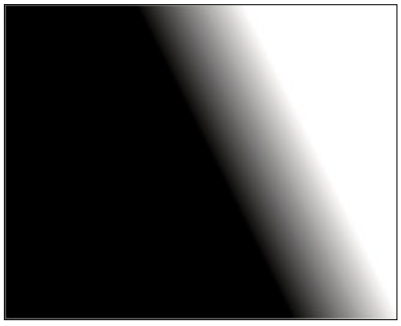
Listing 4: Sinuskurve mit SVG

```
<svg width="1000" height="500" id="sinus"></svg>
<script>
var svg = document.getElementById('sinus');
var axes = {
  x: 100,
  y: 100
};
counter = 0,
x = 0,
y = 180;
var ampl = 80;
var spacing = 3;
var freq = 0.05;
var phase = 20;
for (var i = -100; i < 300; i++) {
  var line = document.createElementNS
  ("http://www.w3.org/2000/svg", "line");
  line.setAttribute('x1', (i - 1) * spacing + axes.x);
  line.setAttribute('y1', Math.sin(freq * (i - 1 +
  phase)) * ampl + axes.y);
  line.setAttribute('x2', i * spacing + axes.x);
  line.setAttribute('y2', Math.sin(freq * (i +
  phase)) * ampl + axes.y);
  line.setAttribute('style',
  "stroke:black;stroke-width:5");
  svg.appendChild(line);
}
</script>
```

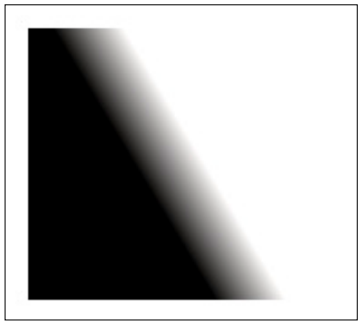
Listing 5: Verläufe

```
<canvas id="canvas" width="600" height="400">
</canvas>
<script>
  var canvas = document.getElementById('canvas');
  var ctx = canvas.getContext("2d");
  if (ctx) {
    ctx.beginPath();
    // Breite der zu zeichnenden Linie
    ctx.lineWidth = 1;
    ctx.strokeStyle = 'rgba(255,255,255,1)';
    ctx.rect(10, 10, 250, 175);
    var gradient = ctx.createLinearGradient(150,
    150, 200, 125);
    gradient.addColorStop(0, "black");
    gradient.addColorStop(0.5, "grey");
    gradient.addColorStop(1, "white");
    ctx.fillStyle = gradient;
    ctx.fill();
    ctx.stroke();
  }
</script>
```

werden Verlaufspunkte in Relation zum Gesamtverlauf gesetzt. Der Parameter *offset* nimmt einen Wert zwischen *0* und *1* an. Der Parameter *color* muss einen CSS-Farbwert darstellen. Bild 9 zeigt das Ergebnis des Beispiels. Ein ähnliches Ergebnis in SVG (Bild 10) wird durch das HTML-Markup in Listing 6 erreicht. Innerhalb des HTML-Tags *<linearGradient id="gradient">* wird der Offset definiert. Das HTML-Tag *<rect>* erzeugt ein Rechteck und erhält über das Attribut *fill="url([gradientID])"* den linearen Verlauf als Füllformat.



Lineare Verlaufsdarstellung, umgesetzt in Canvas (Bild 9)



Die SVG-Variante führt zu einem ähnlichen Ergebnis (Bild 10)

Radialer Verlauf in Canvas

Um einen radialen Verlauf in Canvas zu erzeugen, verwendet man den gleichen Ablauf wie beim linearen Verlauf (Listing 7). Der einzige Unterschied bei der Implementierung ist, dass anstelle der Funktion *createLinearGradient (x0,y0,x1,y1)* die Funktion *createRadialGradient(x1, y1, r1, x2, y2, r2)* verwendet wird. *x0* und *y0* definieren die horizontale und vertikale Position des ersten Kreises, *x1* und *y1* die horizontale und vertikale Position des zweiten Kreises. Der Parameter *r1* beschreibt den Radius des ersten und *r2* den Radius des zweiten Kreises. Bild 11 zeigt das Ergebnis.

Das Gleiche wie für Canvas gilt auch für SVG, jedoch wird dabei zusätzlich das HTML-Tag *<ellipse>* anstelle des *<rect>*-HTML-Tags verwendet, um den radialen Verlauf zu erstellen. Die Attribute *cx* und *cy* des HTML-Tags *<radialGradient>* beschreiben den äußeren und die Attribute *fx* und *fy* den inneren Kreis. Das Attribut *r* beinhaltet den Radius des größten Kreises. Das Ergebnis aus Listing 8 ist in Bild 12 zu sehen.

Zeichnen von Graphen

In den vorangegangenen Abschnitten wurden verschiedene Techniken dargestellt und 2D-Grafiken mit Canvas und SVG vorgestellt. In diesem Abschnitt finden einige der vorgestellten Techniken Verwendung, um die durchschnittlichen Jahreshöchst- und Minimaltemperaturen von Dortmund (Quelle:

Listing 6: Linearer Verlauf

```
<svg height="410" width="800">
  <g>
    <defs>
      <linearGradient id="gradient" x1="0%" y1="100%"
      x2="100%" y2="40%">
        <stop offset="50%" stop-color="black" />
        <stop offset="60%" stop-color="grey" />
        <stop offset="70%" stop-color="white" />
      </linearGradient>
    </defs>

    <!-- The rectangle is filled using a linear
    gradient paint server -->
    <rect fill="url(gradient)"
    stroke="transparent"x="100" y="100" width="150"
    height="150" />
  </g>
</svg>
```

Listing 7: Radialer Verlauf

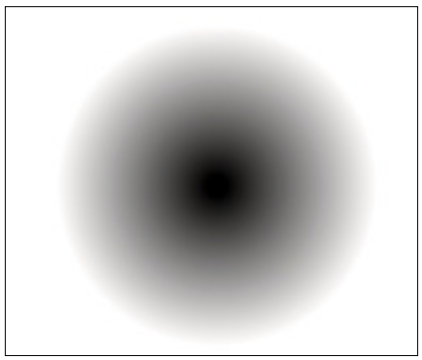
```
<canvas id="canvas" width="600" height="400">
</canvas>

<script>
  var canvas = document.getElementById('canvas');
  var ctx = canvas.getContext("2d");
  if (ctx) {
    ctx.beginPath();
    var gradient = ctx.createRadialGradient(150,
    100, 2, 150, 100, 80);
    gradient.addColorStop(0, "black");
    gradient.addColorStop(0.5, "grey");
    gradient.addColorStop(1, "white");
    ctx.fillStyle = gradient;
    ctx.fillRect(10, 10, 300, 300);
  }
</script>
```

Listing 8: SVG – Radialer Verlauf

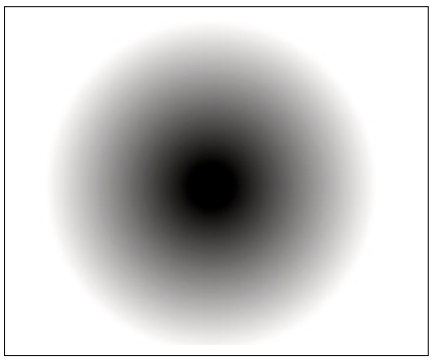
```
<svg width="1000" height="500">
  <defs>
    <radialGradient id="grad1" cx="50%" cy="50%"
    r="50%" fx="50%" fy="50%">
      <stop offset="10%" stop-color="black"
      stop-opacity="1" />
      <stop offset="100%" stop-color="white"
      stop-opacity="1"/>
    </radialGradient>
  </defs>
  <ellipse cx="200" cy="200" rx="85" ry="85"
  fill="url(#grad1)" />
</svg>
```

http://www.wetterkontor.de/de/klima/klima2.asp?land=de&stat=10416) in einem Graphen darzustellen. Die nachfolgenden Listings beschreiben die Umsetzung im Canvas-Umfeld.

Zunächst werden einige Variablen sowie die x- und y-Skala deklariert (Listing 9). Diese Variablen werden im Code genutzt, um die Abstände und Positionen innerhalb des Diagramms zu berechnen. In Listing 10 werden die Skalenbeschriftung und die horizontalen Linien innerhalb des Graphen erstellt. Über jedem Balkendiagrammwert wird zur besseren Lesbarkeit des Diagramms der tatsächliche Wert oberhalb der Balken angezeigt. Listing 11 erstellt mit den Werten aus dem zu Beginn definierten Array *dataName* die Beschriftung der Balken. Daraufhin werden die Balken mit den im Array *maxTemperature* enthaltenen Daten auf das Canvas gezeichnet (Listing 12). Um die Minimaltem- ▶



Canvas: Einen radialen Verlauf in Canvas erzeugen (Bild 11)



SVG: Ein radialer Verlauf, erzeugt mit SVG (Bild 12)

Listing 9: Variablendeklaration

```
<canvas id="canvas" width="600" height="400">
</canvas>
<script src=
"https://code.jquery.com/jquery-2.2.0.min.js">
</script>
<script type="text/javascript">
function calcY(value, minVal, yScalar, canvas) {
  return canvas.height - (value + Math.abs(minVal))
  * yScalar;
}
$(document).ready(function() {
  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");
  var xScalar,yScalar,y;
  var dataName = ["04", "05", "09", "13", "18",
  "21", "22", "22", "19", "15", "09", "05"];
  var maxTemperature = [4, 5, 9, 13, 18, 21, 22, 22,
  19, 15, 9, 5];
  var minTemperature = [-1, -1, 2, 4, 8, 11, 13, 13,
  10, 7, 3, 1];
  var numSamples = 12;
  var maxVal = 40;
  var minVal = -20;
  var stepSize = 10;
  var colHead = 50;
  var rowHead = 60;
  var margin = 10;
  var header = "C°";
  yScalar = (canvas.height - colHead - margin) /
  (maxVal + Math.abs(minVal));
  xScalar = (canvas.width - rowHead) /
  (numSamples + 1);
```

Listing 10: Skalenbeschriftung

```
ctx.beginPath();
  ctx.strokeStyle = "rgba(128,128,255)";
  ctx.font = "14pt Helvetica";
  ctx.fillText(header, 0, colHead - margin);
  ctx.font = "12pt Helvetica";
  var count = 0;
  for (scale = maxVal; scale >= minVal; scale -=
  stepSize) {
    y = colHead + (yScalar * count * stepSize);

    // Zeichnen der Skalenbeschriftung
    ctx.fillText(scale, margin, y + margin);

    // Erstellen eines neuen Referenzpunkts
    ctx.moveTo(rowHead, y);

    // Erstellen eines Pfads für die horizontale
    // Linie
    ctx.lineTo(canvas.width, y);
    count++;
  }

  // Zeichnen der Unterpfade für horizontale Linien
  ctx.stroke();
```

Listing 11: Balkenbeschriftung

```
ctx.stroke();
  ctx.font = "14pt Helvetica";
  ctx.textBaseline = "bottom";
  for (i = 0; i < 12; i++) {
    y = calcY(maxTemperature[i], minVal, yScalar,
    canvas);
    ctx.fillText(dataName[i], xScalar * (i + 1),
    y - margin);
  }
  ctx.stroke();
```

peraturen im Graphen darzustellen, wird in Listing 13 eine rote Linie eingezeichnet, die ihre Daten aus dem Array *minTemperature* bezieht. Das Endergebnis des Programms kann man im Bild 13 betrachten.

Die folgenden Beispiele zeigen die Implementierung in SVG, die ein ähnliches Ergebnis erzielt (Bild 14). Die Implementierung verwendet sowohl HTML-Markup als auch die JavaScript-Implementation. Die Reihenfolge der HTML-Tags ist nur ein Beispiel und kann variieren. Dennoch sollte bei der Implementierung darauf geachtet werden, dass eine klare Struktur besteht, um eine bessere Lesbarkeit für Dritte zu ermöglichen.

In Listing 14 werden die x- und y-Achsen mit dem HTML-Tag <*line*> erstellt. Das Balkendiagramm wird mit <*rect*> implementiert und gleichzeitig, wie im Canvas-Beispiel, oberhalb mit dem lesbaren Wert versehen. Dies wird über das Tag <*text*> erreicht. Am Ende des Listings werden die Achsen mit Hilfe des Tags <*text*> beschriftet.

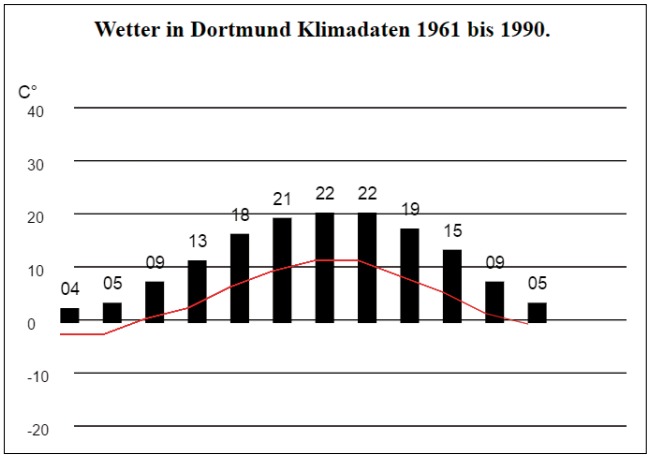
Im Listing 15 kommt ein neues Element zum Einsatz. Es wird über das Array mit den enthaltenen Minimaltemperaturen

Listing 12: Balken

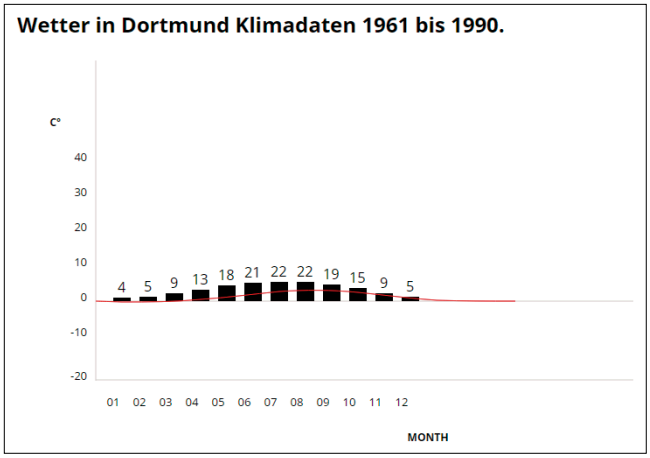
```
var zeroLine = (canvas.height - colHead - margin
- Math.abs(minVal) - maxVal);
for (i = 0; i < 12; i++) {
  y = calcY(maxTemperature[i], minVal, yScalar,
  canvas);
  var height = zeroLine - y;
  ctx.fillRect(xScalar * (i + 1), y, 20, height);
}

var zeroLine = (canvas.height - colHead - margin -
Math.abs(minVal) - maxVal);
for (i = 0; i < 12; i++) {
  y = calcY(maxTemperature[i], minVal, yScalar,
  canvas);
  var height = zeroLine - y;
  ctx.fillRect(xScalar * (i + 1), y, 20, height);
}
```

Listing 13: Minimaltemperaturen

```
ctx.beginPath();
  ctx.strokeStyle = "red";
  ctx.lineWidth = 1;
  var reset = true;
  for (i = 0; i < 12; i++) {
    y = calcY(minTemperature[i], minVal, yScalar,
    canvas);
    if (reset) {
      ctx.moveTo(rowHead, y);
      reset = false;
    }
    ctx.lineTo(xScalar * (i + 1), y);
    ctx.stroke();
  }
});
```



Implementierung der Erstellung eines Graphen in SVG (Bild 13)



Auch mit SVG kann man Graphen erstellen (Bild 14)

Tabelle 1: Vergleich SVG versus Canvas

Canvas	SVG
Pixelbasiert (dynamisches PNG)	Formbasiert
Einzelnes HTML-Element	Mehrere grafische Elemente, die Teil des Dokument-Objekt-Modells (DOM) werden
Nur durch Skript geändert	Durch Skript und CSS geändert
Ereignismodell/Benutzerinteraktion ist präzise *(x,y)*.	Ereignismodell/Benutzerinteraktion ist allgemeiner *(rect, path)*.
Die Leistung ist bei kleineren Oberflächen, einer größeren Objektanzahl (>10T) oder bei beiden besser.	Die Leistung ist bei einer kleineren Objektanzahl (<10T), größeren Oberflächen oder bei beiden besser.

iteriert, das im Kopf des Listings definiert wurde. Pro Eintrag sind drei Werte hinterlegt, was die Verwendung des *path*-Elements ermöglicht. Über das *path*-Element besteht die Möglichkeit, komplexe Formen, Kurven, aber auch Linien zu zeichnen.

Im Grunde genommen besteht die Möglichkeit, alle Grundformen mit dem *path*-Element zu erstellen. Im Klimagraphen findet eine Bézierkurve Verwendung. Im Attribut *d* wird eine Zeichenkette definiert, die aus Befehlen und durch Leerzeichen getrennte Koordinaten besteht.

Der Wert *M* innerhalb des Attributs ist vergleichbar mit *moveTo* von Canvas und setzt den Beginn der Form auf einen neuen Referenzpunkt. *C* impliziert, dass vom aktuellen Referenzpunkt eine Bézierkurve gezeichnet wird. Danach wer- ▶

Listing 14: Achsen

```
<svg version="1.2" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
id="climate" class="graph" aria-labelledby="title"
role="img">
 <title id="title">A line chart showing some
 information</title>
 <g class="grid x-grid" id="xGrid">
  <line x1="90" x2="90" y1="5" y2="371"></line>
 </g>
 <g class="grid y-grid" id="yGrid">
  <line x1="90" x2="705" y1="370" y2="370"></line>
 </g>
 <g class="grid y-grid" id="yGrid">
  <line x1="90" x2="705" y1="280" y2="280"></line>
 </g>
 <g class="bar">
  <rect width="20" height="4" y="276" x="110"></rect>
  <text x="115" y="271" dy="10px">4</text>
 </g>
 <g class="bar">
  <rect width="20" height="5" y="275" x="140"></rect>
  <text x="145" y="270" dy="10px">5</text>
 </g>
 <g class="bar">
  <rect width="20" height="9" y="271" x="170"></rect>
  <text x="175" y="266" dy="10px">9</text>
 </g>
 <g class="bar">
  <rect width="20" height="13" y="267" x="200"></rect>
  <text x="200" y="262" dy="10px">13</text>
 </g>
 <g class="bar">
  <rect width="20" height="18" y="262" x="230"></rect>
  <text x="230" y="257" dy="10px">18</text>
 </g>
 <g class="bar">
  <rect width="20" height="21" y="259" x="260"></rect>
  <text x="260" y="254" dy="10px">21</text>
 </g>
 <g class="bar">
  <rect width="20" height="22" y="258" x="290"></rect>
  <text x="290" y="253" dy="10px">22</text>
 </g>
 <g class="bar">
  <rect width="20" height="22" y="258" x="320"></rect>
  <text x="320" y="253" dy="10px">22</text>
 </g>
 <g class="bar">
  <rect width="20" height="19" y="261" x="350"></rect>
  <text x="350" y="256" dy="10px">19</text>
 </g>
 <g class="bar">
  <rect width="20" height="15" y="265" x="380"></rect>
  <text x="380" y="260" dy="10px">15</text>
 </g>
 <g class="bar">
  <rect width="20" height="9" y="271" x="410"></rect>
  <text x="415" y="266" dy="10px">9</text>
 </g>
 <g class="bar">
  <rect width="20" height="5" y="275" x="440"></rect>
  <text x="445" y="270" dy="10px">5</text>
 </g>
 <g class="labels x-labels">
  <text x="110" y="400">01</text>
  <text x="140" y="400">02</text>
  <text x="170" y="400">03</text>
  <text x="200" y="400">04</text>
  <text x="230" y="400">05</text>
  <text x="260" y="400">06</text>
  <text x="290" y="400">07</text>
  <text x="320" y="400">08</text>
  <text x="350" y="400">09</text>
  <text x="380" y="400">10</text>
  <text x="410" y="400">11</text>
  <text x="440" y="400">12</text>
  <text x="470" y="440" class="label-title">Month
  </text>
 </g>
 <g class="labels y-labels">
  <text x="80" y="120">40</text>
  <text x="80" y="160">30</text>
  <text x="80" y="200">20</text>
  <text x="80" y="240">10</text>
  <text x="80" y="280">0</text>
  <text x="80" y="320">-10</text>
  <text x="80" y="370">-20</text>
  <text x="50" y="80" class="label-title">C°</text>
 </g>
</svg>
```

den drei durch Leerzeichen getrennte Positionsangaben deklariert, um den Fluss der Kurve zu beschreiben.

Einsatzgebiete und Entscheidungsgrundlagen

Auch wenn man mit beiden Techniken fast immer das gleiche Ergebnis erzielen kann, gibt es dennoch Unterschiede und Einsatzgebiete, in denen die eine der anderen Technik vorgezogen werden sollte. Tabelle 1 stellt die Eigenschaften der beiden Techniken gegenüber, anhand derer man Entscheidungen treffen kann.

Optimale Einsatzgebiete für Canvas sind etwa das Rendern von Echtzeitdaten, Spiele, Bild- und Videomanipulationen oder eine hohe Anzahl von Objekten, wie sie zum Beispiel in Heatmaps auftreten können. Optimale Einsatzgebiete von SVG sind dagegen skalierbare Webseiten und Grafiken, die Verwendung von Vektordokumenten, statische Bilder sowie statische Graphen. Beide Techniken stellen ein breitgefächertes Instrumentarium für Webdesigner und Entwickler bereit, mit dem sich optimale Ergebnisse erzielen lassen. ■

Philippe Bénard
ist Senior Software Engineer bei der Adesso AG. Er verfügt über langjährige Berufserfahrung im Bereich der Entwicklung sowie der Analyse von Software-Applikationen.
www.adesso.de

Listing 15: path-Element

```
<script src=
"https://code.jquery.com/jquery-2.2.0.min.js">
</script>
<script>
  $(document).ready(function () {
    var svg = document.getElementById("climate");
    var xSpacing = 30;
    var xCoord = 120;
    var zeroPoint = 280;
    var lastYCoord = 0;
    var yCoord1 = 0;
    var yCoord2 = 0;
    var yCoord3 = 0;
    var climateMin =  [
    { "temperature1": "-1", "temperature2": "-1",
    "temperature3": "2" }',
    { "temperature1": "4", "temperature2": "8",
    "temperature3": "11" }',
    { "temperature1": "13", "temperature2": "13",
    "temperature3": "10" }',
    { "temperature1": "7", "temperature2": "3" ,
    "temperature3": "1" }'];

    var firstEntry = JSON.parse(climateMin[0]);
    var yBeginning = zeroPoint -
    parseInt(firstEntry.temperature1);
    var beginningPath =
    document.createElementNS
    ("http://www.w3.org/2000/svg", "path");
    beginningPath.setAttribute('d',
    "M90 280 C90 280 90 280 120 " + yBeginning);
    beginningPath.setAttribute('style',
    "stroke:red;stroke-width:1;fill:none;");
    svg.appendChild(beginningPath);
    lastYCoord= yBeginning;
    climateMin.forEach(function(e) {
    var climateValue = JSON.parse(e);
    var path = document.createElementNS
    ("http://www.w3.org/2000/svg", "path");
    yCoord1 = zeroPoint -
    parseInt(climateValue.temperature1);
    yCoord2 = zeroPoint -
    parseInt(climateValue.temperature2);
    yCoord3 = zeroPoint -
    parseInt(climateValue.temperature3);
    if (lastYCoord === 0){
    lastYCoord = yCoord3;
    }

    path.setAttribute('d', "M" + xCoord + " " +
    lastYCoord + " C" + (xCoord + xSpacing) + " " +
    (yCoord1) + " " + (xCoord + xSpacing + xSpacing) +
    " " + yCoord2 + " " + (xCoord + xSpacing +
    xSpacing + xSpacing) + " " + yCoord3);
    lastYCoord = yCoord3;
    path.setAttribute('style',
    "stroke:red;stroke-width:1;fill:none;");
    xCoord += xSpacing * 3;
    svg.appendChild(path);
  });

  // Set End
  var endingPath = document.createElementNS
  ("http://www.w3.org/2000/svg", "path");
  endingPath.setAttribute('d', "M" + xCoord + " " +
  lastYCoord + " C" + (xCoord + xSpacing) + " " + 280
  + " " + (xCoord + xSpacing + xSpacing) + " " + 280
  + " " + (xCoord + xSpacing + xSpacing + xSpacing) +
  " " + 280);
  endingPath.setAttribute('style',
  "stroke:red;stroke-width:1;fill:none;");
  svg.appendChild(endingPath);
  });
</script>
```

Developer Newsletter

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

Jetzt kostenlos anmelden:

webundmobile.de

twitter.com/webundmobile

facebook.de/webundmobile

gplus.to/webundmobile

ASPEKTORIENTIERTE PROGRAMMIERUNG IN JAVASCRIPT

Aspekte

Über die aspektorientierte Programmierung lassen sich bestimmte Funktionalitäten vom eigentlichen Anwendungscode trennen.

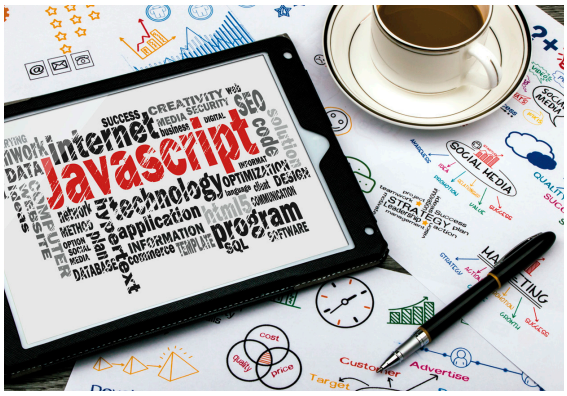


Bild: shutterstock / bleakstar

Bei der aspektorientierten Programmierung (kurz AOP) handelt es sich um ein Programmierparadigma, das im Zusammenhang mit der objektorientierten Programmierung zum Einsatz kommt und dessen Idee es ist, generische Funktionalitäten wie etwa Logging, Caching, Transaktionsmanagement et cetera (sprich sogenannte Cross Cutting Concerns oder im Jargon der aspektorientierten Programmierung auch Aspekte) vom eigentlichen Anwendungscode getrennt zu halten und den Code dadurch insgesamt klarer und modularer zu halten.

Betrachten wir als einfaches Beispiel das Logging. Wer kennt Folgendes nicht: Häufig möchte man wissen, mit welchen Argumenten eine Methode zur Laufzeit aufgerufen wird. Beispielsweise, um in einem Produktivsystem Fehleranalyse zu betreiben oder Ähnliches. Dazu baut man an den Anfang der jeweiligen Methode einen entsprechende Logging-Aufruf ein und versieht diesen vielleicht noch mit einem Feature *Toogle*, um das Logging bei Bedarf an- beziehungsweise wieder abzuschalten. Eventuell greift man dabei auch auf spezielle Logging-Bibliotheken wie winston (https://github.com/winstonjs/winston) zurück, die in dieser Hinsicht deutlich flexibler sind als reine *console.log()*-Aufrufe. Aber all das ändert nichts an der Tatsache, dass der Logging-Code direkt im Anwendungscode steht.

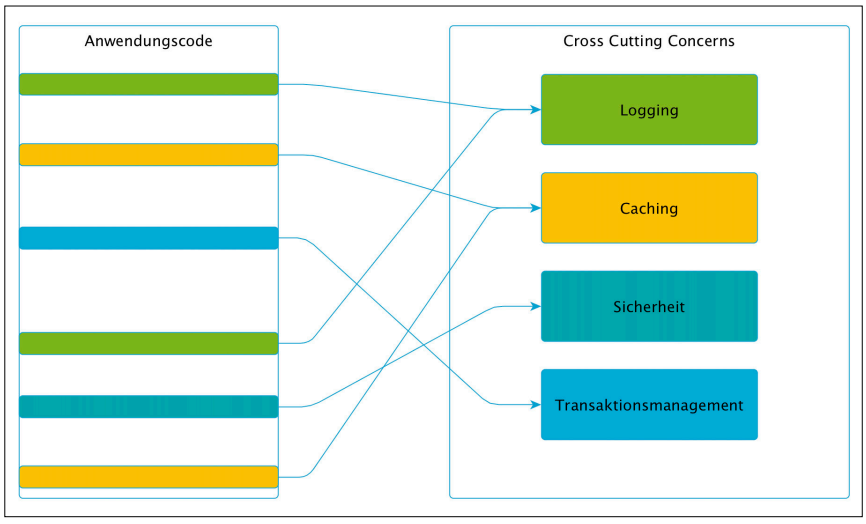
Der Code für den Aspekt Logging ist also wie in Bild 1 skizziert auf mehrere Stellen in der Anwendung verteilt. Bei der AOP dagegen werden wie in Bild 2 skizziert Aspekte wie das Logging getrennt vom eigentlichen Anwendungscode gehalten und über spezielle Techniken mit diesem verknüpft.

Auf diese Weise lassen sich einzelne Aspekte relativ flexibel einzelnen Klassen auch im Nachhinein hinzufügen (oder wieder daraus entfernen), ohne dass die jeweilige Klasse beziehungsweise die entsprechende Stelle im Code an sich geändert werden muss.

Die erste Bibliothek (für Java), die aspektorientierte Programmierung ermöglichte, war AspectJ (https://eclipse.org/aspectj). Ein weiteres prominentes Beispiel ist aber auch etwa das Spring-Framework. Bei diesen und anderen Frameworks werden in der Regel die generischen Funktionalitäten (im Jargon der AOP auch Aspekte genannt) entweder über XML-Dateien oder über Annotationen definiert und konfiguriert und anschließend entweder über einen speziellen Compiler in den endgültigen Code kompiliert oder aber zur Laufzeit dynamisch dazugeladen (in beiden Fällen spricht man auch davon, die Aspekte in den Code zu weben, abgeleitet vom englischen Begriff weaving).

Bevor wir uns anschauen, welche Möglichkeiten es gibt, AOP auch in JavaScript umzusetzen, seien zunächst noch einige Begrifflichkeiten erklärt, die bei AOP verwendet werden (Bild 3):

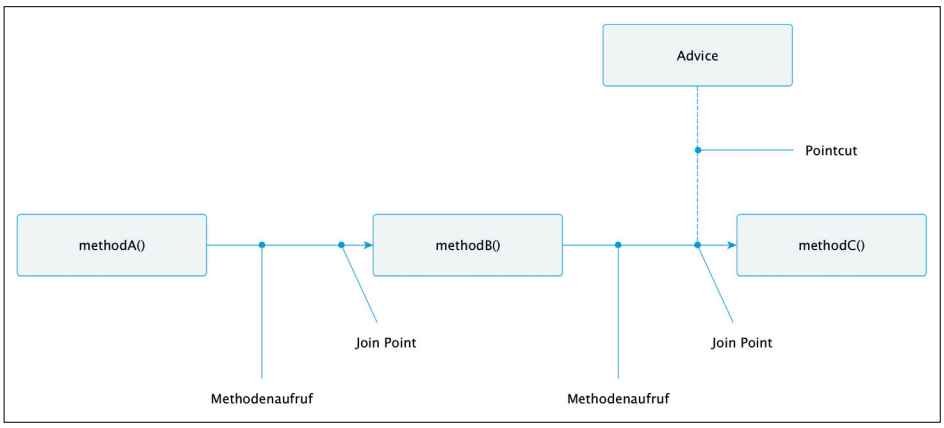
- **Aspect:** Bezeichnet wie schon erwähnt eine generische Funktionalität wie beispielsweise Logging, Caching oder Transaktionsmanagement.
- **Join Point:** Bezeichnet ein Ereignis in der Ausführung eines



Vermischung von Anwendungslogik und Cross Cutting Concerns (Bild 1)

Programms, zum Beispiel das Ausführen eines Code-Blocks, den Aufruf einer Methode, die Initialisierung einer Klasse, das Ausführen eines *catch*-Blocks oder Ähnliches.

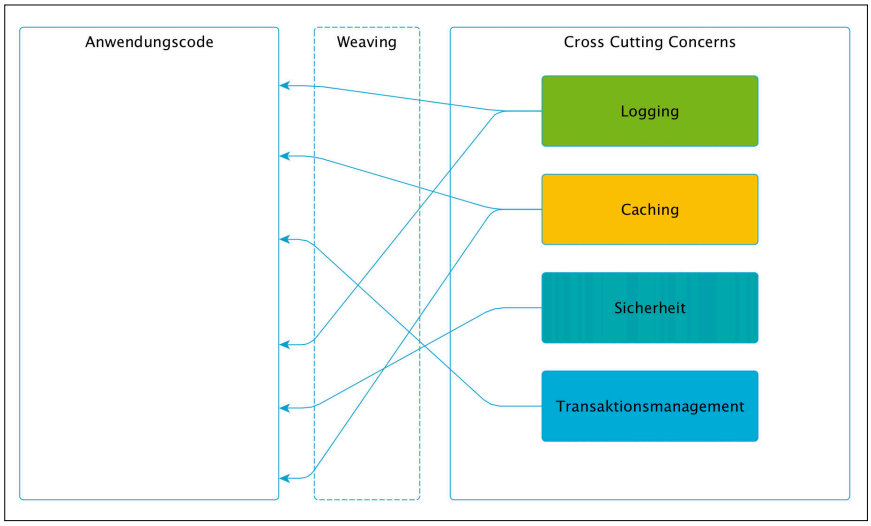
- **Advice:** Bezeichnet die Aktion, die von einem Aspekt ausgeführt werden soll, sobald ein bestimmter Join Point erreicht wurde (beispielsweise: Gebe die Argumente des Methodenaufrufs auf die Konsole aus).
- **Pointcut:** Besteht aus einem oder mehreren Join Points, sprich über Pointcuts lassen sich mehrere Join Points zusammenfassen.



Begrifflichkeiten in der aspektorientierten Programmierung (Bild 3)

Zusätzlich gibt es eine Reihe verschiedener Typen von Advices:

- **Before Advice:** Ein Advice, der vor einem bestimmten Join Point ausgeführt wird und der beispielsweise den Zugriff auf die Argumente eines Methodenaufrufs ermöglicht.
- **After Returning Advice:** Ein Advice, der nach einem bestimmten Join Point ausgeführt wird und der beispielsweise den Zugriff auf den Rückgabewert einer Methode ermöglicht.
- **After Throwing Advice:** Ein Advice, der nach dem Werfen eines Fehlers ausgeführt wird und über den man Zugriff auf den Fehler hat.
- **Around Adivce:** Ein Advice, der um einem bestimmten Join Point herum ausgeführt wird und über den man sowohl Zugriff auf die Argumente einer Methode als auch auf deren Rückgabewert hat.



AOP extrahiert den Code für Cross Cutting Concerns (Bild 2)

Auf diese Weise lassen sich also Aspekte festlegen wie: Immer, wenn die Methode *x()* der Klasse *X()* (= erster Join Point) oder die Methode *y()* der Klasse *Y()* (= zweiter Join Point) aufgerufen wird (= zusammen ein Pointcut), gebe die Argumente auf die Konsole aus (= Advice).

Methoden-Neudefinition

Während es technisch gesehen in anderen Sprachen wie Java nicht ganz trivial ist, AOP zu implementieren (wenn man es von Hand und nicht mit Hilfe eines der genannten Frameworks machen möchte), ist es in JavaScript dank seiner Dynamik deutlich einfacher. Betrachten wir dazu im Folgenden als Beispiel die in Listing 1 gezeigte Klasse *Calculator* mit ▶

Listing 1: Hier werden Anwendungslogik und Logging gemischt

```
'use strict';
class Calculator {
  sum(x, y) {
    console.log('Aufruf von sum() mit den Argumenten
    ${x}, ${y}');
    return x + y;
  }
  prod(x, y) {
    console.log('Aufruf von prod() mit den Argumenten
    ${x}, ${y}');
    return x * y;
  }
}
let calculator = new Calculator();
console.log(calculator.sum(5, 6));
console.log(calculator.prod(5, 6));
```

zwei Methoden *sum()* und *prod()*, welche die Summe beziehungsweise das Produkt zweier Zahlen berechnen. Wie man im Listing sehen kann, sind in beiden Methoden Logging-Befehle enthalten, die eine entsprechende Meldung mit dem Namen der Methode und den jeweils übergebenen Argumenten auf die Konsole ausgeben. Wie bereits beschrieben, ist dies ein typischer Fall der Vermischung von Anwendungslogik und generischer Funktionalität (beziehungsweise einem Cross Cutting Concern).

Im Folgenden möchte ich nun zeigen, mit welchen Techniken sich Aspekte wie das Logging in JavaScript getreu der aspektorientierten Programmierung aus dem eigentlichen Anwendungscode extrahieren lassen.

Dank der Tatsache, dass sich Objektmethoden in JavaScript zur Laufzeit überschreiben beziehungsweise neu definieren lassen, ist die erste Technik relativ naheliegend. Die Idee dabei ist, die Methoden, die mit einem Aspekt verknüpft werden sollen, zu überschreiben und innerhalb der neuen Methoden die Logik für den Aspekt auszuführen sowie die Originalmethode aufzurufen. Ein Beispiel dazu zeigt Listing 2 (wobei man den Code in einer realen Anwendung natürlich auf verschiedene Dateien verteilen würde). Die Methoden *sum()* und *prod()* werden hier jeweils durch neue Funktionen überschrieben, in denen zunächst über Method Borrowing auf die übergebenen Argumente zugegriffen wird, diese dann auf die Konsole ausgegeben werden und anschließend über *apply()* die jeweilige Originalmethode aufgerufen wird.

Der ursprüngliche Code, sprich der Code, in dem die Klasse *Calculator* definiert wird, enthält nun keine Konsolenausgaben mehr.

Allerdings ist der Teil des Codes, der den AOP-Code darstellt, alles andere als kompakt und enthält darüber hinaus – schlimmer noch – doppelten Code. Es ist daher sinnvoll, über eine generische Funktion nachzudenken, die das Verhalten eines Before-Advice implementiert und die beliebige andere Funktionen beziehungsweise Methoden mit einer Dekoratorfunktion (beziehungsweise einem Advice) versieht, die dann vor dem Aufruf der dekorierten Funktion/Methode aufgerufen wird.

Eine solche Funktion *before()* beziehungsweise deren Verwendung ist in Listing 3 zu sehen (siehe auch http://philipsjsrecipes.com/generic-inspect-function-calls). Diese Funktion erwartet eine andere Funktion als Parameter *(decorator)* und gibt eine Funktion zurück, die ihrerseits eine Funktion als Pa-

Listing 2: Einfachste Form der AOP in JavaScript

```
'use strict';
class Calculator {
  sum(x, y) {
    return x + y;
  }
  prod(x, y) {
    return x * y;
  }
}
let calculator = new Calculator();

// Anfang AOP-Code
// 1.) Originalmethoden merken
const originalSum = calculator.sum;
const originalProd = calculator.prod;
// 2.) Methoden überschreiben
calculator.sum = function() {
  let args = Array.prototype.slice.call(arguments);
  // 3.) Advice implementieren, hier: Logging
  console.log('Aufruf von sum() mit den Argumenten
  ${args.join(', ')}');
  // 4.) Aufruf der Originalmethode
  return originalSum.apply(this, arguments);
}
calculator.prod = function() {
  let args = Array.prototype.slice.call(arguments);
  console.log('Aufruf von prod() mit den Argumenten
  ${args.join(', ')}');
  return originalProd.apply(this, arguments);
}
// Ende AOP-Code

console.log(calculator.sum(5, 6));
console.log(calculator.prod(5, 6));
```

Listing 3: Generische before()-Funktion

```
'use strict';
class Calculator {
  sum(x, y) {
    return x + y;
  }
  prod(x, y) {
    return x * y;
  }
}
let calculator = new Calculator();

// Anfang AOP-Code
const before = decorator => fn => function(...args)
{
  decorator(fn, args);
  return fn.apply(this, args);
}
const log = (fn, ...args) => console.log('Aufruf von
${fn.name} mit den Argumenten ${args.join(', ')}');
const logBefore = before(log);
calculator.sum = logBefore(calculator.sum);
calculator.prod = logBefore(calculator.prod);
// Ende AOP-Code

console.log(calculator.sum(5, 6));
console.log(calculator.prod(5, 6));
```

Tabelle 1: Übersicht über die verschiedenen Advices in meld

Advice	Beschreibung
meld.before	Wird vor dem Aufruf einer Methode aufgerufen.
meld.after	Wird nach dem Aufruf einer Methode aufgerufen (unabhängig davon, ob die Methode normal beendet wird oder ob ein Fehler auftritt).
meld.around	Wird um den Aufruf einer Methode herum aufgerufen.
meld.afterReturning	Wird nach dem Aufruf einer Methode aufgerufen, sofern diese ohne Fehler beendet wird.
meld.afterThrowing	Wird beim Auftreten eines Fehlers aufgerufen.

rameter erwartet *(fn)*. Der Aufruf *before(log)* erzeugt demnach eine Funktion, die eine andere Funktion entgegennimmt, dann die *log()*-Funktion aufruft und anschließend die übergebene Funktion.

Die JavaScript-Bibliothek meld

Möchte man nicht alle Arten von Advices selbst implementieren, kann man auch auf Bibliotheken wie meld (https://github.com/cujojs/meld) zurückgreifen, die bereits Implementierungen für Before-Advices, After-Advices et cetera bereitstellen. Die Bibliothek meld kann sowohl im Browser als auch unter Node.js genutzt werden. Im ersteren Fall erfolgt die Installation typischerweise über Yeoman *(yeoman install meld)* oder über Bower *(bower install meld)*, im letzteren Fall über den Node.js Package Manager mit dem Befehl *npm install meld* (im Folgenden wird nur der Einsatz unter Node.js besprochen).

Nach erfolgreicher Installation lässt sich die Bibliothek unter Node.js wie gewohnt über *require('meld')* einbinden. Das so importierte Objekt stellt anschließend verschiedene Methoden zur Verfügung, um einzelne Advices für Objektinstanzen beziehungsweise deren Methoden zu definieren (Tabelle 1). Im Einzelnen sind dies analog zu den eingangs beschriebenen Advice-Typen die Methoden *before()*, *after()*, *around()*, *afterThrowing()* und *afterReturning()*.

Das angepasste Beispiel zur Definition eines Before-Advice zeigt Listing 4. Der Methode *meld.before()* übergibt man als erstes Argument das Objekt, als zweites Argument den Namen der Methode, für die der Advice definiert werden soll, und als drittes Argument eine Funktion, die die Logik enthält, die durch den Advice angestoßen werden soll. Innerhalb dieser Callback-Funktion hat man Zugriff auf die Argumente, mit denen die jeweils dekorierte Methode aufgerufen wird.

Die Definition eines After-Advice, über den man Zugriff auf den Rückgabewert einer Funktion beziehungsweise Methode hat, ist in Listing 5 zu sehen, wobei hier die Methode *after()* verwendet wird. Alternativ können über *afterThrowing()*, und *afterReturning()* aber auch Advices definiert werden, die nur aufgerufen werden, wenn ein Fehler auftritt *(afterThrowing())* oder nur, wenn die Funktion normal beendet wurde *(afterReturning())*. Die Methode *after()* dagegen definiert Advices, die in beiden Fällen aufgerufen werden.

Wie die vorangegangenen Beispiele zeigen, hat man bei einem Before-Advice Zugriff auf die Argumente und bei einem After-Advice (beziehungsweise einem AfterReturning-Advice) Zugriff auf den Rückgabewert einer Methode/Funktion. Es ist jedoch nicht möglich, innerhalb eines After-Advice auf die Argumente oder innerhalb eines Before-Advice auf ▶

Listing 4: Before-Advice

```
'use strict';
const meld = require('meld');
class Calculator {
  sum(x, y) {
    console.log('Berechne ${x} + ${y}');
    return x + y;
  }
}
let calculator = new Calculator();

meld.before(calculator, 'sum', (...args) => {
  console.log('Aufruf von calculator.sum() mit
  Argumenten: ${args.join(', ')}');
});

console.log(calculator.sum(5, 6));
// Aufruf von calculator.sum() mit Argumenten: 5, 6
// Berechne 5 + 6
// 11
```

Listing 5: After-Advice

```
'use strict';
const meld = require('meld');
class Calculator {
  sum(x, y) {
    console.log('Berechne ${x} + ${y}');
    return x + y;
  }
}
let calculator = new Calculator();

meld.after(calculator, 'sum', result => {
  console.log('Aufruf von calculator.sum() ergab:
  ${result}');
});

console.log(calculator.sum(5, 6));
// Berechne 5 + 6
// Aufruf von calculator.sum() ergab: 11
// 11
```

den Rückgabewert zuzugreifen. Benötigt man dennoch beides (also Argumente und Rückgabewert) oder möchte man wissen, zu welchem Zeitpunkt eine Funktion gestartet und zu welchem Zeitpunkt sie beendet wurde (beispielsweise, um die Ausführungsdauer zu berechnen), muss man wie in Listing 6 zu sehen auf den eingangs erwähnten Around-Advice ausweichen. Dieser stellt über seine Callback-Funktion einen Parameter *joinpoint* zur Verfügung, über den man sowohl an die Argumente gelangt (Eigenschaft *args)* als auch an den Rückgabewert (über den Aufruf der Methode *proceed()*, welche die Originalmethode aufruft und deren Ergebnis zurückgibt). Praktisch: Möchte man einen zuvor definierten Advice zu einem späteren Zeitpunkt wieder deaktivieren, reicht es wie in Listing 7 gezeigt, auf dem von den Methoden *before(), after(), around()* et cetera zurückgegebenen Objekt die Methode *remove()* aufzurufen.

Verwendung eines Around-Advice

Ein weiteres Beispiel für die Anwendung von AOP und die Verwendung eines Around-Advice ist die Implementierung einer Caching-Funktionalität: Wird eine Funktion oder Methode mit den gleichen Argumenten aufgerufen, berechnet sie das Ergebnis nicht erneut, sondern liefert das bereits zuvor berechnete Ergebnis aus einem internen Cache zurück.

In Listing 8 ist gezeigt, wie sich die Methode *sum()* mit Hilfe der Bibliothek meld und einer Closure um entsprechende Funktionalität erweitern lässt. Als Cache wird hier ein einfaches Objekt verwendet, dessen Eigenschaften die Schlüssel darstellen und das durch die Closure eingeschlossen wird.

Innerhalb der inneren Funktion wird zunächst der Schlüssel auf Basis der jeweils übergebenen Argumente generiert. Befindet sich für diesen Schlüssel bereits ein Wert im Cache (wurde die Funktion also bereits mit den übergebenen Argumenten aufgerufen), wird dieser Wert zurückgegeben. Existiert dagegen kein Wert für den Schlüssel im Cache, wird über *joinpoint.proceed()* die eigentliche Funktion/Methode aufgerufen und das Ergebnis dieses Aufrufs unter dem Schlüssel in den Cache geschrieben.

Eines der neueren JavaScript-Features, die eventuell in eine der nächsten Versionen des ECMAScript-Standards Einzug halten, sind sogenannte Dekoratoren (beziehungsweise Decorators – siehe https://github.com/wycats/javascript-decorators). Diese Dekoratoren ähneln vom Prinzip (und im Übrigen auch von der Syntax) her Annotationen, wie man sie beispielsweise aus Java kennt, und ermöglichen es, Klassen beziehungsweise Methoden zu markieren.

Dekoratoren werden momentan noch von kaum einer JavaScript-Laufzeitumgebung unterstützt. Es ist jedoch über spezielle Plug-ins möglich, entsprechenden Support für den JavaScript-Transpiler BabelJS (https://babeljs.io) zu aktivie-

Listing 7: Entfernen eines Advice

```
'use strict';
const meld = require('meld');
class Calculator {
  sum(x, y) {
    console.log('Berechne ${x} + ${y}');
    return x + y;
  }
}
let calculator = new Calculator();

const remover = meld.before(calculator, 'sum',
(...args) => {
  console.log('Aufruf von calculator.sum() mit
  Argumenten: ${args.join(', ')}');
});

console.log(calculator.sum(5, 6));
// Aufruf von calculator.sum() mit Argumenten: 5, 6
// Berechne 5 + 6
// 11

remover.remove();
console.log(calculator.sum(5, 6));
// Berechne 5 + 6
// 11
```

Listing 6: Around-Advice zur Ausgabe von Argumenten und Rückgabewert

```
'use strict';
const meld = require('meld');
class Calculator {
  sum(x, y) {
    console.log('Berechne ${x} + ${y}');
    return x + y;
  }
}
let calculator = new Calculator();

meld.around(calculator, 'sum', joinpoint => {
  console.log('Aufruf von calculator.sum() mit
  Argumenten: ${joinpoint.args.join(', ')}');
  let result = joinpoint.proceed();
  console.log('Aufruf von calculator.sum() ergab:
  ${result}');
  return result;
});

console.log(calculator.sum(5, 6));
// Aufruf von calculator.sum() mit Argumenten: 5, 6
// Berechne 5 + 6
// Aufruf von calculator.sum() ergab: 11
// 11
```

ren. Um also die im Folgenden beschriebenen Code-Beispiele (mit Hilfe von Grunt) selbst auszuprobieren, müssen innerhalb des entsprechenden Ordners folgende Befehle ausgeführt und damit *grunt, load-grunt-tasks, grunt-babel, babel-preset-es2015* und *babel-plugin-transform-decorators-legacy* installiert werden (Letzteres übrigens als *legacy* markiert, da Dekoratoren vorher standardmäßig von BabelJS unterstützt wurden, mittlerweile aber aufgrund der noch ausstehenden Übernahme in den ECMAScript-Standard wieder deaktiviert beziehungsweise entfernt wurden):

```
npm install grunt

npm install --save-dev
  load-grunt-tasks
  grunt-babel
  babel-preset-es2015
  babel-plugin-transform-decorators-legacy
```

Ein Dekorator ist zunächst einmal nichts anderes als eine Funktion mit drei Parametern: dem Zielobjekt, auf dem der Dekorator ausgeführt wird, dem Namen beziehungsweise Schlüssel der Eigenschaft des Objekts, für die der Dekorator ausgeführt wird, sowie dem entsprechenden Property-Descriptor dieser Eigenschaft:

Listing 8: Caching über AOP

```
'use strict';
...
meld.around(calculator, 'sum', (() => {
  const cache = {};
  return joinpoint => {
    const key = joinpoint.args.join(', ');
    let result;

    if(!cache[key]) {
      console.log('Berechne Ergebnis für: ${key}');
      cache[key] = joinpoint.proceed();
    } else {
      console.log('Hole Ergebnis aus Cache für:
      ${key}');
    }
    return cache[key];
  }
})());

let result1 = calculator.sum(5, 6);
// --> Berechne Ergebnis für: 5, 6
let result2 = calculator.sum(5, 6);
// --> Hole Ergebnis aus Cache für: 5, 6
let result3 = calculator.sum(8, 9);
// --> Berechne Ergebnis für: 8, 9
let result4 = calculator.sum(8, 9);
// --> Hole Ergebnis aus Cache für: 8, 9
```

```
function decorator(target, key, descriptor) {
  ...
}
```

Optional kann ein Dekorator einen Property-Descriptor zurückgeben, der dann für die entsprechende Eigenschaft verwendet wird. Beispielsweise könnte man relativ einfach einen Dekorator erstellen, der verhindert, dass eine Eigenschaft überschrieben werden kann:

```
function readonly(target, key, descriptor) {
  descriptor.writable = false;
  return descriptor;
}
```

Um eine Methode mit einem Dekorator zu markieren beziehungsweise zu dekorieren, schreibt man den Namen des Dekorators plus einem vorangestelltem @-Zeichen vor die jeweilige Methode (Listing 9).

Die Bibliothek aspect.js

Eine AOP-Bibliothek, die das Konzept der Dekoratoren unterstützt, ist aspect.js (https://github.com/mgechev/aspect.js) von Minko Gechev, dem Autor des Buches »Switching to Angular 2« (übrigens nicht zu verwechseln mit http://aspectjs.com, einer weiteren AOP-Bibliothek für JavaScript).

Installiert werden kann die Bibliothek aspect.js über das Git-Repository:

```
git clone https://github.com/mgechev/aop.js --depth 1
```

Da aspect.js selbst in TypeScript geschrieben ist, müssen zudem ein entsprechender Compiler beziehungsweise eine entsprechende Laufzeitumgebung installiert werden: ▶

Listing 9: Funktionsweise von Dekoratoren

```
'use strict';
function readonly(target, key, descriptor) {
  descriptor.writable = false;
  return descriptor;
}

class Calculator {
  @readonly
  sum(x, y) {
    return x + y;
  }
}

let calculator = new Calculator();
console.log(calculator.sum(5, 6));  // 11
calculator.sum = (x, y) => x - y;
// Nicht möglich, da @readonly
console.log(calculator.sum(5, 6));  // 11
```

```
npm install -g ts-node
npm install -g typescript
```

Die Implementierung des Logging-Beispiels in aspect.js zeigt Listing 10. Einzelne Aspekte werden als separate Klassen definiert (im Beispiel *LoggerAspect)*, wobei man über Dekoratoren innerhalb dieser Aspekt-Klassen die Methoden (beziehungsweise Advices) auszeichnet, die aufgerufen werden sollen. Dabei stellt die Bibliothek aspect.js analog zu den beschriebenen Advices die in Tabelle 2 aufgelisteten Dekoratoren zur Verfügung.

Für welche Klasse(n) beziehungsweise welche Methode(n) ein Advice angewendet werden soll, wird über ein Konfigurationsobjekt gesteuert, das man dem jeweiligen Dekorator als Parameter übergibt: *classNamePattern* beschreibt durch einen regulären Ausdruck die Klasse(n), *methodNamePattern* analog die entsprechende Methode(n).

Tabelle 2: Übersicht über die Advices in aspect.js

Advice	Beschreibung
@beforeMethod (MethodSelector)	Wird vor dem Aufruf einer Methode aufgerufen.
@afterMethod (MethodSelector)	Wird nach dem Aufruf einer Methode aufgerufen.
@aroundMethod (MethodSelector)	Wird um den Aufruf einer Methode herum aufgerufen.
@onThrowOfMethod (MethodSelector)	Wird beim Auftreten eines Fehlers aufgerufen.
@beforeStaticMethod (MethodSelector)	Wird vor dem Aufruf einer statischen Methode aufgerufen.
@afterStaticMethod (MethodSelector)	Wird nach dem Aufruf einer statischen Methode aufgerufen.
@aroundStaticMethod (MethodSelector)	Wird um dem Aufruf einer statischen Methode herum aufgerufen.
@onThrowOfStaticMethod (MethodSelector)	Wird beim Auftreten eines Fehlers bei einer statischen Methode aufgerufen.
@beforeSetter (MemberSelector)	Wird vor dem Aufruf eines Setters aufgerufen.
@afterSetter (MemberSelector)	Wird nach dem Aufruf eines Setters aufgerufen.
@aroundSetter (MemberSelector)	Wird um dem Aufruf eines Setters herum aufgerufen.
@onThrowOfSetter (MemberSelector)	Wird beim Auftreten eines Fehlers bei einem Setter aufgerufen.
@beforeGetter (MemberSelector)	Wird vor dem Aufruf eines Getters aufgerufen.
@afterGetter (MemberSelector)	Wird nach dem Aufruf eines Getters aufgerufen.
@aroundGetter (MemberSelector)	Wird um dem Aufruf eines Getters herum aufgerufen.
@onThrowOfGetter (MemberSelector)	Wird beim Auftreten eines Fehlers bei einem Getter aufgerufen.

Listing 10: Verwendung der Bibliothek aspect.js

```
import {beforeMethod, Wove, Metadata} from
'../aop.js/lib/aspect';

class LoggerAspect {
  @beforeMethod({
    classNamePattern: /^Calculator/,
    methodNamePattern: /^sum/
  })
  invokeBeforeMethod(meta: Metadata) {
    console.log(
      'Aufruf von ${meta.className}.
      ${meta.method.name}() mit Argumenten:
      ${meta.method.args.join(', ')}'
    );
  }
}

@Wove()
class Calculator {
  sum(x: number, y: number) {
    console.log('Berechne ${x} + ${y}');
    return x + y;
  }
}
let calculator = new Calculator();
console.log(calculator.sum(5, 6));
// Aufruf von Calculator.sum() mit Argumenten: 5, 6
// Berechne 5 + 6
// 11
```

Fazit

Die aspektorientierte Programmierung (AOP) ermöglicht es, Anwendungscode mit zusätzlicher Funktionalität zu erweitern, ohne diesen verändern zu müssen. Mit anderen Worten: Mit Hilfe von AOP kann Anwendungscode frei von Cross Cutting Concerns gehalten werden. In JavaScript gibt es verschiedene Möglichkeiten, AOP umzusetzen: Überschreiben von Methoden und die Verwendung von Dekoratoren. ■

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

DDC .NET Developer Conference 2016

Trends, Lösungen und Know-how für Profi-Entwickler

05. – 07.12.2016, Köln

web & mobile DEVELOPER-Leser erhalten 15% Rabatt mit Code DDC16wmd

DevSessions	Konferenz	Workshops
05.12.2016	**06.12.2016**	**07.12.2016**
8 halbtägige Workshops	1 Keynote, 12 Vorträge	Ganztägiges Praxistraining

Ihre Experten (u.a.):

Bernd Marquardt
Freier Consultant und Autor

Dr. Holger Schwichtenberg
Fachlicher Leiter IT-Visions.de

Gernot Starke
Gründungsmitglied des iSAQB e.V.

David Tielke
Trainer & Berater david-tielke.de

Ralf Westphal
Mitgründer der Clean Code Developer Initiative

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

Partner: proximity TECHNOLOGY, Saxonia Systems So geht Software.

Präsentiert von: dotnetpro

Veranstalter: developer media, Neue Mediengesellschaft Ulm mbH

INTERNATIONALIZATION API

Sprachabhängig

Das Internationalization API ermöglicht eine sprachabhängige Formatierung.

Sobald eine Webseite oder Webanwendung mehrsprachig sein soll, ist eine vollständige Internationalisierung (kurz i18n für das i und das letzte n sowie die dazwischen liegenden 18 Buchstaben in dem englischen Wort internationalization) ebensolcher unumgänglich. Die Idee dabei ist – so viel dürfte bekannt sein –, eine Anwendung so zu strukturieren, dass sie ohne großen Aufwand an verschiedene Sprachen (oder noch allgemeiner: an verschiedene Regionen) angepasst werden kann.

Textinhalte in verschiedenen Sprachen

In der Regel sind die Textinhalte einer Anwendung dabei für jede Sprache (beziehungsweise für jedes Locale, dazu gleich mehr) jeweils in einer separaten Datei enthalten und werden dann je nach Nutzereinstellungen beziehungsweise Spracheinstellungen geladen.

Dies ist vom Prinzip her innerhalb einer Anwendung relativ einfach zu implementieren, wenn man beispielsweise Templating-Engines verwendet, die die einzelnen Textbausteine aus diesen Wörterbuchdateien beziehen.

Etwas komplexer ist da schon die Handhabung verschiedener Konventionen, was die Formatierung von Zahlenwerten, Datums- und Zeitangaben oder den Vergleich von Zeichenketten angeht. In Deutschland steht bei Datumsangaben bekanntermaßen zuerst der Tag, dann der Monat, dann das Jahr, jeweils mit Punkten voneinander getrennt (zum Beispiel 12.08.2016). In den USA dagegen steht der Monatsname an erster Position, gefolgt von Tag und Jahr (zum Beispiel 08/12/2016), wobei das Slash-Symbol als Trennzeichen verwendet wird.

In Großbritannien ist es dagegen von der Anordnung her wie in Deutschland, allerdings wird hier wie bei der amerikanischen Schreibweise das Slash-Symbol als Trennzeichen verwendet: 12/08/2016.

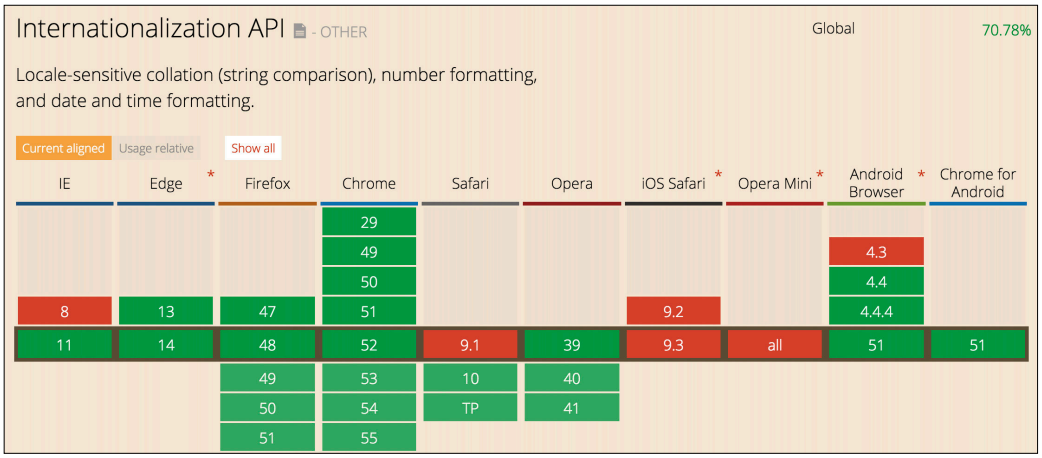
Bei Zahlenwerten oder Währungsangaben sieht es ähnlich aus: Während in Deutschland als Tausendertrennzeichen der Punkt verwendet wird, als Trennzeichen der Nachkommastellen das Komma und das Währungssymbol hinter dem Zahlenwert steht (beispielsweise 123.456,79 €), ist es in den USA genau andersherum: Als Tausendertrennzeichen wird hier das Komma verwendet, als Nachkommatrennzeichen der Punkt, und das Währungssymbol steht vor dem Zahlenwert (€123,456.79). Und in anderen Teilen der Welt sieht es wiederum anders aus. Da kann es verständlicherweise schnell zu Verwirrung kommen, wenn man diese ganzen Regeln selbst beachten und implementieren müsste.

Bevor wir uns nun im Detail mit dem Internationalization API beschäftigen, im Folgenden zunächst ein paar grundlegende Informationen und Begriffserklärungen zum Thema Internationalisierung.

Gebietsschemaparameter

Das Festlegen von Sprache beziehungsweise einer Region, in der eine bestimmte Sprache gesprochen wird, geschieht, wie eben schon kurz erwähnt, über sogenannte Locales. Laut Wikipedia handelt es sich dabei um einen Einstellungssatz, der die Gebietsschemaparameter (Standortparameter) für Computerprogramme enthält, oder anders gesagt um einen Identifier (auch Language Tag genannt), der genau definiert, welche Sprache verwendet werden soll, und demnach auch, wie Zahlenwerte, Datums- und Zeitangaben formatiert und Zeichenketten verglichen werden sollen.

Der Aufbau eines Language Tags ist im IETF-Dokument BCP47 (https://tools.ietf.org/html/bcp47) spezifiziert: Generell besteht ein Language Tag aus durch Minuszeichen ge-



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

trennten Buchstaben- und Zahlenkombinationen (diese werden auch Sub Tags genannt).

Das erste Sub Tag ist das Language Sub Tag, das entweder einen aus zwei Zeichen bestehenden Code aus dem ISO-Standard 639-1 (https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes) enthalten darf oder einen aus drei Zeichen bestehenden Code aus den ISO-Standards 639-2 (https://en.wikipedia.org/wiki/List_of_ISO_639-2_codes), 639-3 (https://en.wikipedia.org/wiki/List_of_ISO_639-3_codes) und 639-5 (https://en.wikipedia.org/wiki/List_of_ISO_639-5_codes). So lautet das Locale beispielsweise für Deutsch *de*, das für Englisch *en*, das für Italienisch *it* et cetera.

Alle weiteren Sub Tags sind optional und dienen ausschließlich dazu, die Sprache weiter einzugrenzen. Über das Region Sub Tag lässt sich beispielsweise das Land beziehungsweise die Region definieren, in der die Sprache gesprochen wird: Das Locale *de-DE* beispielsweise definiert Deutsch, wie es in Deutschland gesprochen wird; das Locale *de-AT* dagegen, wie es in Österreich gesprochen wird, das Locale *en-US* definiert Englisch, wie es in den USA gesprochen wird, und das Locale *en-UK* Englisch, wie es in Großbritannien gesprochen wird.

Schriftsystem definieren

Über das sogenannte Script Sub Tag ist es möglich, das zu verwendende Schriftsystem zu definieren. Für viele Sprachen – wie beispielsweise Englisch – ist dies zwar überflüssig, weil hier nur ein Schriftsystem verwendet werden kann. Allerdings gibt es auch Sprachen wie Chinesisch, die verschiedene Schriftsysteme erlauben.

Mögliche Werte für das Script Sub Tag sind in ISO 15924 definiert und können unter https://de.wikipedia.org/wiki/ISO_15924 nachgelesen werden. Das Locale *de-Latn* definiert beispielsweise Deutsch in lateinischer Schrift, das Locale *de-Latf* dagegen Deutsch in Fraktur, das Locale *zh-Hans* definiert Chinesisch in vereinfachter Schrift und das Locale *zh-Hant* Chinesisch in traditioneller Schrift.

Über das Variant Sub Tag ist es darüber hinaus möglich, bestimmte Varianten einer Sprache zu definieren. Beispielsweise repräsentiert das Locale *de-CH-1996* die Sprache Deutsch in der Schweiz nach der Rechtschreibreform von 1996.

Man sieht: Es gibt quasi keine Sprachvariante, die man über die verschiedenen Sub Tags nicht abdecken könnte. Doch damit nicht genug: Über sogenannte Extension Sub Tags lassen sich Locale-Definitionen nahezu beliebig erweitern. Für JavaScript ist in diesem Zusammenhang die in RFC 6067 definierte BCP 47 Extension U (https://tools.ietf.org/html/rfc6067) von Bedeutung.

Den Namen hat die Erweiterung aufgrund des Zeichens *u*, das hierbei bei der Definition eines Sub Tags vorangestellt wird, gefolgt von einem Schlüssel und einem Wert. Das Locale *de-DE-u-co-phonebk* beispielsweise definiert die Sprache Deutsch, gesprochen in Deutschland, wobei als Sortierreihenfolge *(co* für collation) die Sortierung wie im deutschen Telefonbuch *(phonebk* als Abkürzung für phonebook) verwendet wird. Weitere Möglichkeiten, die sich über die BCP 47 Extension U ergeben, sind die Definition des zu verwendenden Kalenders (über den Schlüssel *ca* für calendar, also beispielsweise *de-DE-u-ca-buddhist)*, der zu verwendenden Währung (über den Schlüssel *cu* für currency), des zu verwendenden Zahlensystems (über den Schlüssel *nu* für numeric system) und der zu verwendenden Zeitzone (über den Schlüssel *tz* für timezone).

Listing 1: Support überprüfen

```
if (window.Intl && typeof window.Intl === 'object')
{
  // Internationalization API wird unterstützt
} else {
  // Internationalization API wird nicht unterstützt
}
```

Das Internationalization API

Die eingangs in diesem Artikel geschilderte Problematik der verschiedenen Formatierungen ist genau der Punkt, an dem das ECMAScript 2017 Internationalization API (https://tc39.github.io/ecma402) ansetzt: Es ermöglicht die sprachabhängige Formatierung von Zahlenwerten, Datums- und Zeitangaben sowie den sprachabhängigen Vergleich von Zeichenketten.

Das API orientiert sich dabei an bestehenden Internationalization APIs wie denen des .NET Frameworks oder des Java Internationalization API. Die erste Version des API wurde vom ECMA TC39 bereits 2010 ins Leben gerufen. Mittlerweile ist das API in der vierten Version erschienen und wird von den meisten aktuellen Browsern unterstützt (Bild 1).

Für ältere Browser können Polyfills wie zum Beispiel Intl.js genutzt werden (https://github.com/andyearnshaw/Intl.js). Ob ein Browser das API unterstützt, kann mit dem Code in Listing 1 überprüft werden.

Alle Objekte rund um die Internationalization API werden über das zentrale Objekt *Intl* bereitgestellt. Dieses Objekt stellt analog zu den eingangs genannten Anforderungen drei Typen zur Verfügung:

- *Intl.Collator:* Ermöglicht den Vergleich von Zeichenketten unter Berücksichtigung von Internationalisierungsinformationen.
- *Intl.DateTimeFormat:* Ermöglicht die Formatierung von Datums- und Zeitangaben unter Berücksichtigung von Internationalisierungsinformationen.
- *Intl.NumberFormat:* Ermöglicht die Formatierung von Zahlenwerten unter Berücksichtigung von Internationalisierungsinformationen.

Konkrete Instanzen dieser Typen können über die entsprechenden Konstruktorfunktionen erzeugt werden, wobei in allen Fällen das Locale als Zeichenkette übergeben wird:

```
'use strict';
const locale = 'de-DE';
```

```
const dateTimeFormatGermany = new
Intl.DateTimeFormat(locale);
const numberFormatGermany = new
Intl.NumberFormat(locale);
const collactorGermany = new Intl.Collator(locale);
```

Für den Fall, dass man das Locale weglässt, wird das vom Nutzer standardmäßig verwendete Locale verwendet:

```
'use strict';

const dateTimeFormatGermany = new Intl.DateTimeFormat();
const numberFormatGermany = new Intl.NumberFormat();
const collactorGermany = new Intl.Collator();
```

Alternativ ist es auch möglich, statt eines einzelnen Locale ein Array von Locales zu übergeben. In diesem Fall wird das erste Locale im Array verwendet, das vom System des Nutzers unterstützt wird.

Dabei werden für den Fall, dass ein Locale, das nicht unterstützt wird, aus mehreren Sub Tags besteht, die Sub Tags sukzessive aus dem Locale entfernt und mit dem jeweils entstandenen Locale erneut geprüft. Werden alle im Array enthaltenen Locales nicht unterstützt, wird das Standard-Locale verwendet.

```
'use strict';
const locales = [
  'zh-Hant', // Erst wird zh-Hant überprüft, dann zh
  'en-US',   // ... dann en-US, dann en, ...
  'de-DE'    // ... dann de-DE, dann de.
];

const dateTimeFormatGermany = new
Intl.DateTimeFormat(locales);
const numberFormatGermany = new
Intl.NumberFormat(locales);
const collactorGermany = new
Intl.Collator(locales);
```

Zudem kann allen Konstruktorfunktionen als zweiter Parameter ein Konfigurationsobjekt übergeben werden, über das sich weitere Aspekte steuern lassen.

Listing 2: Sortierung mit Intl.Collator

```
'use strict';
const names = [
  'Mustermann, Max',
  'Müller, Max',
  'Mustermann, Moritz',
  'Mueller, Moritz',
  'Meier, Petra',
  'Meier, Peter'
];
const log = message => console.log(message);
const collator = (locale, options) =>
new Intl.Collator(locale, options);
const nameCollator = collator('de-DE', {usage:
'sort'});
log(names.sort(nameCollator.compare));
// [
// Meier, Peter,
// Meier, Petra,
// Mueller, Moritz,
// Müller, Max,
// Mustermann, Max,
// Mustermann, Moritz
//]
```

Listing 3: Sortierung wie im Telefonbuch

```
...
const phonebookCollator = collator(
  'de-DE-u-co-phonebk', {usage: 'sort'}
);
log(names.sort(phonebookCollator.compare));
// [
// Meier, Peter,
// Meier, Petra,
// Müller, Max,
// Mueller, Moritz,
// Mustermann, Max,
// Mustermann, Moritz
//]
```

Unterstützung von Locale

Ob ein Locale unterstützt wird, lässt sich übrigens über die statischen Methoden *Intl.Collator.supportedLocalesOf(), Intl.DateTimeFormat.supportedLocalesOf()* und *Intl.NumberFormat.supportedLocalesOf()* herausfinden. Jede dieser Methoden erwartet als Argument entweder ein einzelnes Locale oder ein Array von Locales und liefert als Rückgabewert ein Array mit denjenigen der übergebenen Locales, die unterstützt werden:

```
'use strict';

const locales = ['zh-Hant', 'en-US', 'de-DE'];
console.log(Intl.Collator.supportedLocalesOf(locales));
console.log(Intl.DateTimeFormat.
supportedLocalesOf(locales));
console.log(Intl.NumberFormat.
supportedLocalesOf(locales));
```

Neben den konkret im Internationalization API definierten Typen und deren Methoden gibt es an den Standardobjekten Array, String, Number und Date folgende Methoden, die zwar nicht Teil dieses API sind, jedoch ebenfalls entsprechende Locale-Informationen berücksichtigen: *Array.prototype.*

Tabelle 1: Konfiguration von Intl.Collator

Eigenschaft	Beschreibung
localeMatcher	Der zu verwendende Matching-Algorithmus.
usage	Angabe darüber, ob der Vergleich durch den Collator für die Sortierung von oder für die Suche nach Zeichenketten verwendet werden soll. Mögliche Werte sind entsprechend *sort* und *search*.
sensitivity	Angabe darüber, welche Zeichen als ungleich erachtet werden sollen. Mögliche Werte sind *base*, *accent*, *case* und *variant*.
ignorePunctuation	Boolesche Angabe darüber, ob Satzzeichen ignoriert werden sollen.
numeric	Boolesche Angabe darüber, ob Zeichenketten numerisch verglichen werden sollen (beispielsweise 1 < 5 < 10).
caseFirst	Boolesche Angabe darüber, ob Kleinbuchstaben oder Großbuchstaben in der Sortierung vorne stehen.

Listing 4: Formatierung von Datums- und Zeitangaben

```
'use strict';
const date = new Date(Date.UTC(2016, 8, 15, 8, 0,
0));
const log = message => console.log(message);
const dateTimeFormat = (locale, options) =>
  new Intl.DateTimeFormat(locale, options);

log(dateTimeFormat('de').format(date));
// 15.9.2016
log(dateTimeFormat('en-US').format(date));
// 9/15/2016
log(dateTimeFormat('en-GB').format(date));
// 15/09/2016
log(date.toLocaleString('de'));
// 15.9.2016, 10:00:00
log(date.toLocaleString('en-US'));
// 9/15/2016, 10:00:00 AM
log(date.toLocaleString('en-GB'));
// 15/09/2016, 10:00:00
log(date.toLocaleDateString('de'));
// 15.9.2016
log(date.toLocaleDateString('en-US'));
// 9/15/2016
log(date.toLocaleDateString('en-GB'));
// 15/09/2016
log(date.toLocaleTimeString('de'));
// 10:00:00
log(date.toLocaleTimeString('en-US'));
// 10:00:00 AM
log(date.toLocaleTimeString('en-GB'));
// 10:00:00
```

toLocaleString, *String.prototype.localeCompare*, *String.prototype.toLocaleLowerCase*, *String.prototype.toLocaleUpperCase*, *Number.prototype.toLocaleString*, *Date.prototype.toLocaleString*, *Date.prototype.toLocaleDateString* und *Date.prototype.toLocaleTimeString*.

Vergleich von Zeichenketten

Für den Vergleich von Zeichenketten dient der Typ *Collator*. Dazu stellt er die Methode *compare()* zur Verfügung, die zwei Zeichenketten als Parameter erwartet und einen von drei Zahlenwerten zurückgibt: *1* für den Fall, dass die erste Zeichenkette größer als die zweite ist, sprich in der Sortierung hinter der zweiten Zeichenkette angeordnet wird, *-1* für den Fall, dass die zweite Zeichenkette größer als die erste ist, und *0* für den Fall, dass beide Zeichenketten gleich sind:

```
'use strict';

const nameCollator = new Intl.Collator('de-DE');
console.log(
  nameCollator.compare('Mustermann', 'Meier')
);  // 1
console.log(
  nameCollator.compare('Meier', 'Mustermann')
);  // -1
console.log(
  nameCollator.compare('Meier', 'Meier')
);  // 0
```

Die Art und Weise, wie zwei Zeichenketten miteinander verglichen werden, beziehungsweise welche Regeln hierfür gelten sollen, kann über die Eigenschaft *sensitivity* des Konfigurationsobjekts, das der Konstruktorfunktion *Collator* übergeben werden kann, weiter angepasst werden. Der Wert *base* beispielsweise sorgt dafür, dass alle Zeichen, die die gleiche Basis haben (zum Beispiel die Buchstaben á, a und A), als gleichwertig interpretiert werden:

```
'use strict';

const nameCollator = new Intl.Collator('de-DE');
const nameCollatorBase= new Intl.Collator('de-DE',
  {
    sensitivity: 'base'
  }
);
console.log(nameCollator.compare('Mueller',
'mueller'));// 1
console.log(nameCollatorBase.compare('Mueller',
'mueller')); // 0
```

Die *compare()*-Methode lässt sich praktischerweise auch für die Sortierung von Arrays einsetzen. Den entsprechenden Code zeigt Listing 2. Zur Erinnerung: Der Array-Methode *sort()* kann optional eine Sortier- beziehungsweise Vergleichsfunktion übergeben werden, die dann die Elemente im jeweiligen Array paarweise miteinander vergleicht und ▶

Listing 5: Formatierung von Datums- und Zeitangaben

```
'use strict';
...

log(dateTimeFormat('en-US', {
  year: '2-digit',
  month: '2-digit'
}).format(date)); // 09/16

log(dateTimeFormat('en-US', {
  year: '2-digit',
  month: '2-digit',
  day: '2-digit',
  hour: '2-digit',
  minute: '2-digit',
  second: '2-digit'
}).format(date)); // 09/15/16, 10:00:00 AM

log(dateTimeFormat('en-US', {
  weekday: 'long',
  era: 'long',
  year: '2-digit',
  month: '2-digit',
  day: '2-digit',
  hour: '2-digit',
  minute: '2-digit',
  second: '2-digit',
  timeZoneName: 'long'
}).format(date));

// Thursday, 09 15, 16 Anno Domini,
// 10:00:00 AM Central European Summer Time

log(dateTimeFormat('de', {
  weekday: 'long',
  era: 'long',
  year: '2-digit',
  month: '2-digit',
  day: '2-digit',
  hour: '2-digit',
  minute: '2-digit',
  second: '2-digit',
  timeZoneName: 'long'
}).format(date));

// Donnerstag, 15. 09 16 n. Chr.,
// 10:00:00 Mitteleuropäische Sommerzeit

const formatter = dateTimeFormat('de', {
  weekday: 'long',
  era: 'long',
  year: '2-digit',
  month: '2-digit',
  day: '2-digit',
  hour: '2-digit',
  minute: '2-digit',
  second: '2-digit',
  timeZoneName: 'long'
});

log(formatter.resolvedOptions());
/*
{
  locale: de,
  numberingSystem: latn,
  calendar: gregory,
  timeZone: Europe/Berlin,
  timeZoneName: short,
  era: long,
  year: 2-digit,
  month: 2-digit,
  day: 2-digit,
  weekday: long,
  hour12: false,
  hour: 2-digit,
  minute: 2-digit,
  second: 2-digit
}
*/
```

somit eine Sortierung herstellt. Übergibt man hierbei die *compare()*-Methode des jeweiligen Collators, kann dieser einem die Sortierarbeit abnehmen.

Über den Kennzeichner *u* können, wie erwähnt, innerhalb eines Locale Unicode-Extensions definiert werden, wobei jeweils Schlüssel-Wert-Paare anzugeben sind. Das in Listing 3 verwendete Locale *de-DE-u-co-phonebk* beispielsweise verwendet die Extension *co* mit dem Wert *phonebk*, wodurch eine Sortierung wie im Telefonbuch erreicht wird, sprich ä wird wie ae interpretiert, ö wie oe und ü wie ue.

Eine Übersicht aller Eigenschaften, die an dem Konfigurationsobjekt für Collator-Objekte gesetzt werden können, zeigt Tabelle 1.

Der Typ *DateTimeFormat* ermöglicht, wie bereits erwähnt, die Formatierung von Datums- und Zeitangaben. Dazu stellt er die Methode *format()* zur Verfügung, die eine Objektinstanz von Date als Parameter erwartet und eine den Konfigurationen entsprechende Zeichenkette zurückgibt, die das Datumsobjekt repräsentiert (Listing 4, Listing 5).

Formatierung von Datums- und Zeitangaben

Eine Auflistung der Konfigurationsmöglichkeiten zeigt Tabelle 2. Alternativ zu der Verwendung von *DateTimeFormat* können die Methoden *toLocaleString()*, *toLocaleDateString()* und *toLocaleTimeString()* direkt an der entsprechenden Date-Objektinstanz verwendet werden, wobei ebenfalls die Locale-

Tabelle 2: Konfiguration von Intl.DateTimeFormat

Eigenschaft	Beschreibung
localeMatcher	Der zu verwendende Matching-Algorithmus.
timeZone	Die zu verwendende Zeitzone.
hour12	Boolesche Angabe darüber, ob 12 Stunden oder 24 Stunden als Basis zugrunde gelegt werden sollen.
formatMatcher	Der zu verwendende Matching-Algorithmus.
weekday	Angabe zur Formatierung des Wochentags. Mögliche Werte sind *narrow*, *short* und *long*.
era	Angabe zur Formatierung des Zeitalters. Mögliche Werte sind *narrow*, *short* und *long*.
year	Angabe zur Formatierung des Jahres. Mögliche Werte sind *numeric* und *2-digit*.
month	Angabe zur Formatierung des Monats. Mögliche Werte sind *numeric*, *2-digit*, *narrow* und *short*.
day	Angabe zur Formatierung des Tages. Mögliche Werte sind *numeric* und *2-digit*.
hour	Angabe zur Formatierung der Stunden. Mögliche Werte sind *numeric* und *2-digit*.
minute	Angabe zur Formatierung der Minuten. Mögliche Werte sind *numeric* und *2-digit*.
second	Angabe zur Formatierung der Sekunden. Mögliche Werte sind *numeric* und *2-digit*.
timeZoneName	Angabe zur Formatierung der Zeitzone. Mögliche Werte sind *short* und *long*.

Informationen berücksichtigt werden, die man den Methodenaufrufen übergibt.

Die Formatierung von Zahlenwerten funktioniert vom Prinzip her ähnlich wie die Formatierung von Datums- und Zeitangaben: Auch der Konstruktor von *NumberFormat* erwartet als ersten Parameter das Locale sowie optional als zweiten Parameter ein Konfigurationsobjekt zur weiteren Verfeinerung der Formatierung.

So kann zum Beispiel über *useGrouping* definiert werden, ob Trennzeichen (wie beispielsweise Tausendertrennzeichen) verwendet werden sollen, über *currency* die Währung definiert werden (für den Fall, dass die Zahl als Geldbetrag formatiert werden soll) und über *minimumFractionDigits* und *maximumFractionDigits* definiert werden, wie viele Nachkommastellen eine Zahl mindestens oder maximal haben soll. Eine vollständige Übersicht der Konfigurationsmöglichkeiten finden Sie in Tabelle 3.

Fazit

Das Internationalization API vereinfacht die komplexeren Themen der Internationalisierung, sei es den Vergleich und die Sortierung von Zeichenketten oder die formatierte Ausgabe von Zahlenwerten, Datums- und Zeitangaben. Das API wird mittlerweile von dem Großteil der moderneren Browser unterstützt. Für ältere Browser stehen Polyfill-Bibliotheken zur Verfügung. ■

Tabelle 3: Konfiguration von Intl.NumberFormat

Eigenschaft	Beschreibung
localeMatcher	Der zu verwendende Matching-Algorithmus.
style	Der zu verwendende Formatierungsstil. Zur Auswahl stehen *decimal* für normale Zahleformatierung, *currency* für Währungsformatierung und *percent* für Prozentangaben.
currency	Die bei der Währungsformatierung zu verwendende Währung, beispielsweise *EUR* für Euro oder *USD* für US-Dollar.
currencyDisplay	Angabe darüber, wie bei der Währungsformatierung die Währung dargestellt werden soll. Mögliche Werte sind *symbol* für die Verwendung von Symbolen, *code* für den entsprechenden ISO-Währungscode oder *name* für den Namen der Währung (Euro, Dollar).
useGrouping	Boolesche Angabe darüber, ob Trennzeichen (wie beispielsweise Tausendertrennzeichen) verwendet werden sollen (beispielsweise 1.000.000).
minimumIntegerDigits	Minimale Anzahl der Stellen, die eine Zahl bei der Formatierung haben soll. Verfügt eine Zahl nicht über diese Mindestanzahl, wird – von vorne beginnend – mit Nullen aufgefüllt.
minimumFractionDigits	Minimale Anzahl der Nachkommastellen, die eine Zahl bei der Formatierung haben soll. Verfügt eine Zahl nicht über diese Mindestanzahl, wird – von der letzten Ziffer ausgehend – mit Nullen aufgefüllt.
maximumFractionDigits	Maximale Anzahl der Nachkommastellen, die eine Zahl bei der Formatierung haben soll. Verfügt eine Zahl über mehr Stellen, wird entsprechend gerundet.
minimumSignificantDigits	Minimale Anzahl der signifikanten Stellen, die eine Zahl bei der Formatierung haben soll. Verfügt eine Zahl nicht über diese Mindestanzahl, wird – von der letzten Ziffer ausgehend – mit Nullen aufgefüllt.
maximumSignificantDigits	Maximale Anzahl der signifikanten Stellen, die eine Zahl bei der Formatierung haben soll. Verfügt eine Zahl über mehr signifikante Stellen, wird entsprechend gerundet.

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

ELEMENT QUERIES

Anpassungsfähig

Wenn die klassischen Media Queries nicht ausreichen, helfen Element Queries weiter.

Media Queries gelten im Allgemeinen als wichtigste Komponente für responsive Webseiten. Dank Media Queries lassen sich eigene CSS-Angaben für verschiedene Viewportgrößen machen. So können Sie unterschiedliche Formatierungen für große wie für kleine Browserfenster durchführen. Klassischerweise definiert man etwa ein mehrspaltiges Layout bei viel verfügbarem Platz und ein einspaltiges für kleine Bildschirme. Das funktioniert sehr gut für das grundlegende Layout, schwieriger wird es aber für kleinere Komponenten des Layouts: Für deren optimale Anzeige sind oft nicht die Viewportgrößen ausschlaggebend, sondern die sie direkt umgebenden Elemente.

Zweimal dieselbe Tabelle: Je nachdem, wie breit sie ist, wird eine andere Darstellung benötigt (Bild 1)

Nehmen wir als Beispiel eine Tabelle. Inzwischen gibt es eine Reihe von Lösungen, um Tabellen im responsiven Webdesign gut darzustellen. So kann man eine aufgelöste Darstellung untereinander bei kleinen Viewports wählen und die klassische Tabellendarstellung, sofern genügend Platz vorhanden ist. Was aber tut man jetzt, wenn bei einem Projekt die Tabelle auf manchen Unterseiten im großen Hauptbereich angezeigt werden soll und auf anderen hingegen in der kleineren Sidebar (Bild 1)? In diesem Fall kommen wir mit Media Queries nicht weiter, weil die zu ändernde Darstellung nicht von der Viewportgröße abhängt.

Genau das ist ein typischer Fall für Element Queries. Element Queries sind schon länger im Gespräch – und inzwischen gibt es eine schöne JavaScript-Bibliothek namens EQCSS, die es erlaubt, heute schon Element Queries einzusetzen.

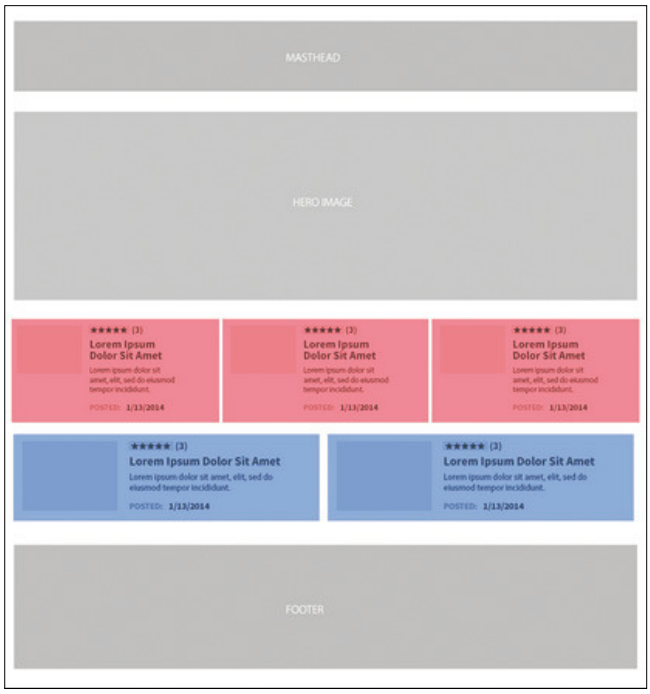
Einsatzbereiche von Element Queries

Sehen wir uns mögliche Einsatzbereiche für Element Queries genauer an. Die Grenzen von Media Queries offenbaren sich zum einen bei modularen Komponenten und zum anderen bei flexiblen Inhalten. Beginnen wir mit typischen Fällen von Komponenten.

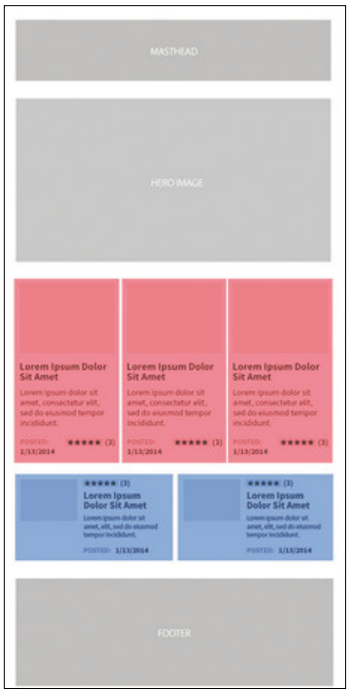
Webseiten bestehen oft aus kleineren Bestandteilen, die in unterschiedlichen Zusammenhängen auftauchen. Gute UI-Komponenten sollten so gestaltet sein, dass sie bis zu einem gewissen Grad in unterschiedlichen Kontexten funktionieren.

Ein Login-Formular kann die zentrale Komponente auf der Hauptlogin-Seite sein, aber auch als kleinere unauffälligere Komponente auf allen Seiten in der Seitenleiste vorhanden sein.

Ein anderes Beispiel ist eine Produktpräsentation, deren Inhalte unterschiedlich angeordnet werden – mal ist das Foto oben und der Text unterhalb, mal sind Bild und



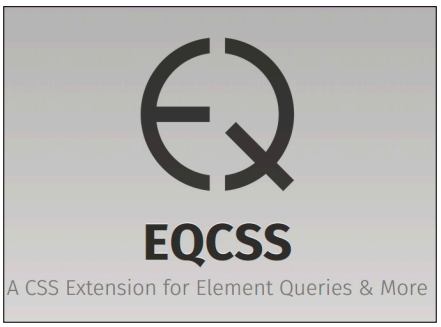
Typischer Fall für Element Queries: Elemente müssen unterschiedlich dargestellt werden, je nachdem, wie viel Platz sie haben (Bild 2)



Die verschiedenen Darstellungen hängen aber nicht vom Viewport ab (Bild 3)

Text nebeneinander. Welche Variante zur Darstellung gewählt wird, hängt dabei nicht nur vom Viewport ab, sondern auch von der Breite, die das Element aktuell hat (Bild 2, Bild 3).

Natürlich lässt sich das ebenso mit herkömmlichen CSS bewerkstelligen, aber es ist einigermaßen aufwendig und viel Code muss doppelt geschrieben werden – wie beispielsweise »Use Cases and Requirements for Element Queries« (https://responsiveimagescg.github.io/cq-usecases) zeigt.



EQCSS: Die Homepage des Projekts (Bild 4)

Aber Element Queries brauchen wir nicht nur bei modularen Komponenten, sondern auch für flexible Inhalte. Normalerweise ist die Inhaltsmenge nicht genau vorgegeben. Beim Einsatz eines Content-Management-Systems (CMS) werden die Inhalte aktualisiert und können dann im Ergebnis auch unterschiedlich lang sein. Eine Überschrift kann einmal aus einigen wenigen Wörtern bestehen und im anderen Fall aus einem ganzen Satz. In diesem Fall könnte man sich natürlich behelfen, indem man Vorgaben für die Länge solcher Elemente macht.

Probleme bei mehrsprachigen Webseiten

Diese Vorgaben funktionieren jedoch nicht bei mehrsprachigen Webseiten, bei denen Navigationspunkte, Menübeschriftungen et cetera von Haus aus unterschiedlich viele Zeichen haben. Schließlich wären dann da noch die von Besuchern oder Benutzern erstellten Inhalte – beispielsweise bei einem Portal für die Vermietung von Ferienwohnungen, wo die Vermieter selbst die Beschreibungen ihrer Objekte eingeben.

In all diesen Fällen haben wir unterschiedliche Inhaltsmengen. Hier wäre es wünschenswert, dass das Design entsprechend darauf reagiert. Beispielsweise könnte die Überschrift in einer kleineren Schriftgröße angezeigt werden, wenn sie länger ist. Speziell für Menüs könnte es sinnvoll sein, dass die modifizierte Anzeige, wie sie für kleinere Screens vorgesehen ist, zum Beispiel bei Sprachen mit längeren Beschriftungen früher ausgelöst wird.

Alle Situationen, in denen wir Layoutveränderungen brauchen, die nicht mit der Viewportgröße zusammenhängen, sind Einsatzbereiche für Element Queries.

Element Queries sind in CSS pur derzeit nicht möglich, aber es gibt mit EQCSS (Bild 4) eine elegante Möglichkeit, sie einzusetzen, dank tatkräftiger Mitwirkung per JavaScript.

Eigenschaften von EQCSS

EQCSS hat eine Reihe von überzeugenden Eigenschaften:

- Die Syntax ist so nah wie möglich an CSS angelehnt. Das heißt, sie ist intuitiv und erlaubt einen schnellen Einstieg.
- Außerdem basiert EQCSS nicht auf einer Bibliothek wie jQuery, das heißt, es gibt keinen Overload, wenn man bei einer Webseite bewusst auf jQuery verzichtet.
- EQCSS bietet sogar noch eine Unterstützung für den Internet Explorer 8 – bemerkenswert angesichts der Tatsache, dass dieser Browser nicht einmal Media Queries versteht.
- Veröffentlicht ist EQCSS unter der MIT-Lizenz.

Um EQCSS einzusetzen, müssen Sie es zuerst einmal herunterladen. Das geht über Git:

```
git clone https://github.com/eqcss/
eqcss.git
```

Oder mit NPM:

```
npm install eqcss
```

Und Sie können die JavaScript-Datei natürlich auch direkt von der Git-Projektseite herunterladen. Danach binden Sie sie ein:

```
<script src="EQCSS.js"></script>
```

Alternativ steht das Skript auch per CDN unter https://cdnjs.cloudflare.com/ajax/libs/eqcss/1.2.1/EQCSS.min.js zur Verfügung.

Sofern Sie noch den IE < 9 unterstützen möchten, schreiben Sie zusätzlich einen Verweis auf das Polyfill innerhalb von konditionalen Kommentaren:

```
<!--[if lt IE 9]>
<script src="EQCSS-polyfills.js"></script>
<![endif]-->
```

Damit sind die Vorbereitungen abgeschlossen. Sie können nun Element Queries sowohl in internen als auch in externen Stylesheets einsetzen. Es kann sinnvoll sein, den EQCSS-Code nicht mit dem normalen CSS-Code zu vermischen, sondern ihn in eine eigene Datei zu schreiben. Und das funktioniert folgendermaßen: Benutzen Sie zum Einbinden das *script*-Element mit der Pseudo-MimeType-Angabe *text/eqcss*. Geben Sie außerdem der Datei die Endung *eqcss*:

```
<script type="text/eqcss" src=styles.eqcss></script>
```

EQCSS wird beim Laden der Seite tätig oder wenn sich die Größe des Browserfensters verändert. Wo nötig, können Sie die Aktionen von EQCSS so auch direkt anstoßen:

```
EQCSS.apply();
```

Das ist sinnvoll, wenn sich die Inhalte dynamisch verändern.

Der Aufbau der Element Queries folgt dem von Media Queries. Eine typische Media Query sieht folgendermaßen aus: Zuerst steht *@media* mit Bedingungen. Von geschweiften Klammern werden die Formatierungen für die angegebene Viewportgröße umfasst:

```
@media screen and (min-width:20em) {
  body {
```

```
    font-size: 80%;
  }
}
```

Ganz ähnlich sehen Element Queries bei EQCSS aus: Auch hier setzen Sie eine @-Regel ein, dieses Mal lautet sie *@element*. Dahinter folgt ein Selektor, um das Element auszuwählen, dessen Eigenschaften abgefragt werden sollen. Der Selektor steht also dort, wo bei Media Queries die Angabe für das Ausgabemedium steht:

```
@element '.container' {
}
```

Wichtig ist dabei, dass der Selektor hinter *@element* in Anführungszeichen angegeben ist.

Im obigen Beispiel werden Elemente mit der Klasse *.container* ausgewählt. Innerhalb der geschweiften Klammern geben Sie dann den CSS-Code an. Beachten Sie, dass Sie innerhalb dieses Blocks noch einen Selektor mit den gewünschten Angaben schreiben müssen. Das Folgende funktioniert also nicht:

```
@element '.container' {
  background: red;
}
```

So hingegen klappt es:

```
@element '.container' {
  h2 {
    background: red;
  }
}
```

Damit werden alle *h2*-Elemente rot eingefärbt, sofern es ein Element mit der Klasse *.container* gibt.

Meist werden Sie jedoch nicht nur auf das Vorhandensein von bestimmten Elementen prüfen, sondern auf ausgewählte Eigenschaften der Elemente reagieren. So können Sie Formatierungen von der Breite eines Elements abhängig machen:

```
@element '.container' and (min-width: 500px) {
  body {
    background: red;
  }
}
```

Im Beispiel wird *body* rot, wenn *.container* mindestens 500 px breit ist.

Elementeigenschaften

Welche Eigenschaften von Elementen lassen sich abfragen? Erwartungsgemäß kann man natürlich auf die Ausmaße von Elementen reagieren. Das heißt, man kann die von Media Queries bekannten Eigenschaften nutzen wie Mindest- und Maximalbreite *(min-width* und *max-width)* sowie Mindest- und Maximalhöhe *(min-height* und *max-height)*.

Aber es stehen weitere Eigenschaften zur Verfügung, die kein Äquivalent bei Media Queries besitzen:

- *min-character* und *max-character*: Die Anzahl an Zeichen. So können Sie beispielsweise die Schriftgröße herabsetzen, wenn eine bestimmte Anzahl an Zeichen in einem Element steht.
- *min-lines* und *max-lines*: Die Anzahl der Zeilen. Die Zeilenanzahl könnte ebenfalls ein sinnvolles Kriterium sein, um die Schriftgröße zu verändern oder ähnliche Anpassungen durchzuführen.
- *min-children* und *max-children*: Die Anzahl der Kindelemente. Auf den ersten Blick könnte man meinen, das sei nicht notwendig, weil wir doch schon quantitative Queries haben. Mit den sogenannten quantitativen Queries können Sie natürlich auch auf die Anzahl an Elementen reagieren. Der Unterschied besteht aber darin, dass Sie bei den quantitativen Queries immer nur die Elemente selbst oder ihre Nachfahren für Sonderformatierungen auswählen können. Bei den Element Queries jedoch können Sie, sofern die Bedingung zutrifft, beliebige Elemente der Webseite ansprechen und anders gestalten.
- *min-scroll-y, max-scroll-y, min-scroll-x, max-scroll-y*: Minimale oder maximale Scrollposition in der Horizontalen (y) oder in der Vertikalen (x). Damit lassen sich besondere Formatierungen definieren, wenn der Benutzer bereits x Pixel oder x Prozent gescrollt hat.

Verschiedene Eigenschaften sind gleichzeitig abfragbar. So können Sie für das Anpassen einer Überschrift auf die Anzahl der Zeichen und den eingenommenen Platz reagieren.

```
@element 'h1' and (min-characters: 16) and (max-width:
500px) {
  /* CSS-Code */
}
```

Nehmen wir an, Ihr Dokument enthält zwei Absätze mit der Klasse *.benutzer* und mit unterschiedlichen Inhalten:

```
<p class="benutzer">Hadschi Halef Omar</p>
<p class="benutzer">Hadschi Halef Omar Ben Hadschi Abul
Abbas Ibn</p>
```

Wir wollen jetzt die Schrift verkleinern, wenn die Absätze mindestens 20 Zeichen haben. Im ersten Ansatz würde man dafür den folgenden Code benutzen:

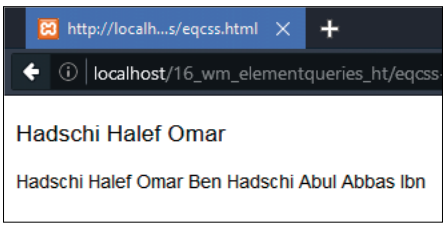
```
@element ".benutzer" and (min-characters: 20) {
  .benutzer {
    font-size: 80%;
  }
}
```

Das funktioniert allerdings nicht wie gewünscht. Im Beispiel werden beide Absätze in kleinerer Schrift dargestellt. Der

Grund dafür ist, dass die allgemeine Bedingung zutrifft:

```
@element ".benutzer" and
(min-characters: 20)
```

Es gibt im Dokument ja ein Element mit der Klasse *.benutzer*, das mindestens 20 Zeichen hat. Und damit werden alle *.benutzer*-Elemente ausgewählt und angepasst.



Bei mehr Zeichen wird die Schrift automatisch verkleinert (Bild 5)

Wir wollen jetzt aber tatsächlich nur bei den Absätzen mit mehr als 20 Zeichen die geänderte Formatierung umsetzen. Dafür brauchen wir den Metaselektor *$this*. Als Metaselektoren werden selektorähnliche Ausdrücke bezeichnet, die es nicht im klassischen CSS gibt, sondern nur in EQCSS.

Das Beispiel lässt sich mit *$this* folgendermaßen umformulieren:

```
@element ".benutzer" and (min-characters: 20) {
  $this {
    font-size: 80%;
  }
}
```

Dadurch wird jetzt nur das *.benutzer*-Element mit mehr als 20 Zeichen in kleinerer Schrift dargestellt (Bild 5).

Weitere Metaselektoren

Immer wieder gibt es Situationen, in denen ein bestimmter Selektor zutrifft und man das zugehörige Elternelement auswählen möchte. Das geht leider mit klassischen CSS-Mitteln nicht – aber es funktioniert bei EQCSS: Hier wählen Sie das Elternelement über den Metaselektor *$parent* aus:

```
@element '.container' {
  $parent {
    background: red;
  }
}
```

Dadurch wird das Elternelement des *.container*-Elements rot eingefärbt. Die Metaselektoren lassen sich mit normalen Selektoren kombinieren und wie normale einfache Selektoren benutzen.

```
@element ".benutzer" and (min-characters: 20) {
  $this h2 {
    font-size: 80%;
  }
}
```

In diesem Beispiel wählt *$this h2* nur das innerhalb des angesprochenen Elements stehende *h2*-Element aus.

Neben den Metaselektoren *$this* und *$parent* gibt es noch *$root* für das Wurzelelement sowie *$prev* und *$next* für vorhergehendes und nachfolgendes Element.

Falls Sie mit Sass arbeiten, würde die Schreibweise der Metaselektoren Probleme bereiten. Glücklicherweise existiert jedoch auch eine Sass-kompatible Notation: Dafür ersetzen Sie das Dollarzeichen durch *eq_* – statt *$parent* schreiben Sie also *eq_parent*.

JavaScript-Logik in CSS

Als weitere Ergänzung von EQCSS können Sie nun innerhalb von CSS JavaScript-Code ausführen. Dafür gibt es *eval()*. Im folgenden Beispiel wird *eval()* benutzt, um über erzeugte Inhalte die jeweils aktuelle Jahreszahl in den Footer zu schreiben.

```
@element 'footer' {
  $this:after {
    content: "© eval('new Date().getFullYear()')";
  }
}
```

Das ist nur ein Beispiel für die Funktionsweise von *eval()*. In der Praxis wäre es natürlich besser, das Datum als normalen Text innerhalb von HTML anzugeben.

Wenn Sie sich fragen, wie EQCSS eigentlich funktioniert, ist ein Blick in die Entwickler-Tools aufschlussreich: Sie sehen dann, dass den Elementen dynamisch *data*-Attribute zugewiesen werden, wie hier im Beispiel:

```
<body data-eqcss-0-0-parent="" data-eqcss-0-1-parent="">
<p class="benutzer" data-eqcss-0-0="" data-eqcss-0-1-
prev=""> Hadschi Halef Omar</p>
<p class="benutzer" data-eqcss-0-0-next=""data-eqcss-0-1
=""> Hadschi Halef Omar Ben Hadschi Abul Abbas Ibn</p>
```

Diese erhalten dann die gewünschten Formatierungen. Das Praktische an EQCSS ist aber, dass Sie selbst nicht mit *data*-Attributen innerhalb des HTML-Parts arbeiten müssen, sondern Ihren CSS-Code ganz wie gewohnt schreiben können.

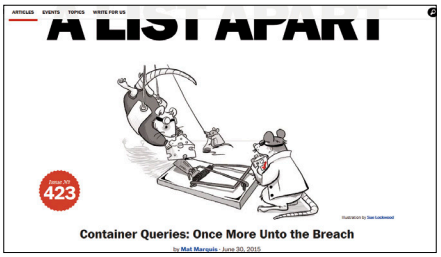
Probleme bei EQ

Ein prinzipielles Problem bei Element Queries ist, dass sich rekursive Angaben erstellen lassen – also Endlosschleifen. Endlosschleifen gibt es in allen Programmiersprachen, aber das klassische CSS ist davor geschützt. Ein Beispiel demonstriert der folgende Code:

```
.container {
  width: 400px;
  background: red;
}
@element '.container' and (min-width: 300px) {
  $this {
    width: 200px;
    background: blue;
  }
}
```

▶

In diesem Fall hat das Element zuerst eine Breite von 400 px. Dann kommt eine Element Query, die definiert, dass bei einer Mindestbreite des Elements von 300 px die Breite auf 200 px gesetzt. Das trifft zu und wird ausgeführt. Jetzt ist das Element 200 px breit. Damit trifft die in der Element Query formulierte Bedingung aber nicht mehr zu, das Element darf also nicht mehr auf 200 px gesetzt werden … wenn es wieder seine ursprüngliche Breite hat, trifft die Bedingung hingegen wieder zu und es wird auf 200 px gesetzt et cetera.



Vorstellung von Container Queries (Bild 6)

EQCSS ist hier pragmatisch: Die zuletzt definierte Regel gilt, das heißt, das Element bleibt 200 px breit und ist blau. Genauso pragmatisch verfährt EQCSS auch bei sich widersprechenden Element Queries, wie im folgenden Beispiel:

```
@element '.widget' and (min-width: 400px) {
  $this {
    width: 200px;
  }
}
@element '.widget' and (max-width: 300px) {
  $this {
    width: 500px;
  }
}
```

In diesem Fall geht EQCSS die Angaben von oben nach unten durch und die letzte setzt sich durch – es ist demnach die Reihenfolge entscheidend. Besser ist es sicherlich, so etwas zu vermeiden. Man könnte sich selbst etwa die Regel auferlegen, zumindest nie die Eigenschaft zu ändern, die in der Element Query abgefragt wird. Aber natürlich gibt es auch Wechselwirkungen zwischen verschiedenen Eigenschaften. Und es liegt es in der Natur von CSS, dass man bei komplexen Stylesheets leicht den Überblick darüber verliert, welche Bedingungen noch gelten – und deswegen können solche Endlosschleifen immer wieder mal passieren.

Alternative Ansätze

Es gibt mehrere Ansätze für die Nachbildung von Element Queries per JavaScript. Die Gefahr, rekursive Angaben zu machen, ist oft das Totschlagargument gegen Element Queries. Mat Marquis von der ursprünglichen Arbeitsgruppe zu den responsiven Bildern regt deswegen in einem Artikel bei A list apart an, sich von Element Queries zu verabschieden und stattdessen Container Queries, wie er sie nennt, einzusetzen (Bild 6). Das Entscheidende an seinem Konzept ist, dass vom Design her Abfragen nach Eigenschaften eines Elements nur die Nachkommen des Elements beeinflussen können – nicht das Element selbst. Der Vorteil: Das Risiko von Endlosschleifen ist gebannt. Die Syntax, die Mat Marquis vorschlägt, sieht so aus:

```
.mod:media( min-width: 320px ) .mod-cost {
  float: left;
}
```

In diesem Fall wird – wenn das Element mit der Klasse *mod* eine Mindestbreite von 320 px hat – das Nachfahrenelement *.mod-cost* links gefloatet.

Ein anderer Vorschlag namens CSS Element Queries stammt von Marc J. Schmidt. Hier werden die Bedingungen in Form von Attributselektoren angegeben.

```
.widget-name h2 {
  font-size: 12px;
}

.widget-name[min-width~="400px"] h2 {
  font-size: 18px;
}

.widget-name[min-width~="600px"] h2 {
  padding: 55px;
  text-align: center;
  font-size: 24px;
}
```

Der Vorteil: Die Syntax ist normales CSS und JavaScript sorgt dafür, dass die entsprechenden Attribute bei den Elementen ergänzt werden. Allerdings kann man damit natürlich nur das Element selbst oder seine Nachfahren formatieren – EQCSS ist hier freier. Derzeit gibt es nur die Eigenschaften *width/*

Links zum Thema

- EQCSS
  http://elementqueries.com
  https://www.smashingmagazine.com/2016/07/how-i-ended-up-with-element-queries-and-how-you-can-use-them-today
- Anwendungsfälle für Element Queries
  https://responsiveimagescg.github.io/cq-usecases
- Quantitative Queries
  http://alistapart.com/article/quantity-queries-for-css
- Container Queries
  http://alistapart.com/article/container-queries-once-more-unto-the-breach
- Weitere Lösungen für Element Queries
  http://marcj.github.io/css-element-queries (CSS Element Queries)
  https://github.com/Snugug/eq.js (EQ.js)
  https://github.com/filamentgroup/elementary (Elementary)

height mit den *min-* und *max*-Präfixen. Beim Vorschlag EQ.js von Sam Richards geben Sie den Code bei den Elementen selbst an (was Sie aber auch über JavaScript machen können):

```
<div class="component" data-eq-pts="small: 400,
medium: 600, large: 900">
  <h1>Hello World</h1>
</div>
```

Diese Elemente lassen sich dann im CSS-Code ansprechen:

```
.container[data-eq-state$="small"],
.container[data-eq-state$="medium"],
.container[data-eq-state$="large"] {
  font-size: 1em;
}

.container[data-eq-state$="small"] {
  border-color: green;
  background-color: rgba(0, 128, 0, 0.25);
}
```

Eine weitere Lösung stammt von Scott Jehl von der Filamentgroup und nennt sich Elementary. Seine Lösung setzt auf Pseudoelemente. Wir haben folgendes Element:

```
<p class="mod mod-foo">...</p>
```

Um das Element mit potenziellen Element-Query-Breakpoints für Mindestbreiten zu versehen, müssen Sie diese bei *:before* angeben – die Zahlen werden dabei durch Leerzeichen getrennt hintereinander geschrieben:

```
.mod-foo:before {
  content: "300 450 620";
}
```

Damit kümmert sich das Skript bei allen Elementen mit der *.mod*-Klasse um passende *data-minwidth*-Attribute, die automatisch je nach Situation erzeugt werden. Diese Attribute wiederum lassen sich mit CSS auswählen. Im folgenden Beispiel erhält das Element bei einer Mindestbreite von 450 px eine blaue Hintergrundfarbe:

```
.mod[data-minwidth~="450"] {
  background: blue;
}
```

Wenn man die verschiedenen alternativen Ansätze mit EQCSS vergleicht, so überzeugt EQCSS zum einen dadurch, dass beliebige Elemente formatiert werden können, und zum anderen verfügt es über mehr abfragbare Eigenschaften. Ein weiteres Plus ist die angenehme Syntax, die sich wie nativ anfühlt.

Apropos nativ: Natürlich gibt es auch in HTML/CSS teilweise direkt Möglichkeiten, Elemente an ihre Umgebung anzupassen. Das heißt, es gibt Anpassungen jenseits von Media Queries. So können Sie bei responsiven Bildern unterschiedliche Bilder nicht nur je nach Viewportgröße, sondern auch abhängig vom Layout ausliefern. Ein Beispiel zeigt der folgende Code:

```
<img src="small.jpg"
  srcset="large.jpg 1024w, medium.jpg 640w,
  small.jpg 320w"
  sizes="(min-width: 36em) 33.3vw, 100vw"
  alt="A rad wolf">
```

Bei *sizes* verrät man dem Browser, dass ab einer Mindestbreite des Viewports von *36 em* das Element nur noch einen Drittel des Viewports einnimmt.

Eine weitere automatische Anpassung gibt es bei dem Multicolumn-Layout-Modul. Hier können Sie über *column-width* die bevorzugte Spaltenbreite festlegen. Je nachdem, wie viel Platz zur Verfügung steht, werden dann die Inhalte auf mehrere Spalten verteilt. Entscheidend ist dabei der verfügbare Platz des Elements selbst, nicht der Viewport.

Automatische Anpassungen zeigen sich auch bei gefloateten Elementen, denn diese werden bei zu wenig Platz automatisch untereinander dargestellt. Das ist beispielsweise praktisch für Anordnungen bei Bildergalerien.

Beim neuen Layoutmodul Flexbox ist eine automatische Aufteilung auf Spalten oder Zeilen dank der Angabe *flex-wrap: wrap* möglich.

Das sind aber alles Anpassungen im Kleineren, die überhaupt nicht die Optionen bieten, die für Element Queries gefordert werden und die mit EQCSS zur Verfügung stehen.

Fazit

EQCSS überzeugt. Die Syntax ist gut entworfen, weil sie intuitiv und einleuchtend ist. Praktisch ist auch, dass die abfragbaren Eigenschaften nicht auf Ausmaße beschränkt sind, sondern beispielsweise auch den Inhalt betreffen – wie die Zeichen- oder Zeilenzahl.

Trotzdem machen Element Queries die klassischen Media Queries nicht überflüssig: Diese sind nach wie vor die beste Möglichkeit, Anpassungen aufgrund des verfügbaren Platzes im Viewports durchzuführen.

Komponenten zu gestalten, die dann überall platziert werden können und sich immer an die Umgebung anpassen – das ist ein schöner Traum, dem wir mit Element Queries einen guten Schritt näher kommen. ■

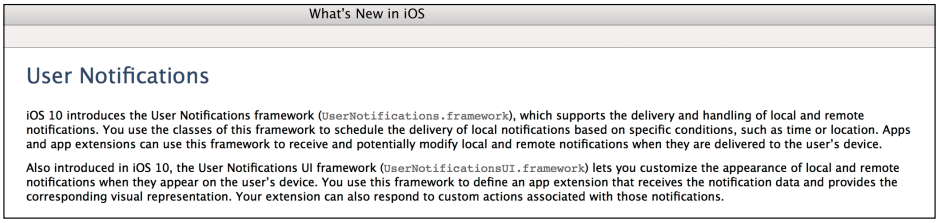
Dr. Florence Maurice
ist Autorin, Trainerin und Programmiererin in München. Sie schreibt Bücher zu PHP und CSS3 und gibt Trainings per Video. Außerdem bloggt sie zu Webthemen unter:
http://maurice-web.de/blog

USERNOTIFICATIONS-FRAMEWORK

Neue Notifications

Zum Erstellen von Notifications stellt Apple ab iOS 10 ein neues Framework bereit.

Auf der WWDC 2016 gab es für Apple-Entwickler wieder zahlreiche Neuheiten zu bestaunen. Alle vier großen Plattformen von Apple – iOS, macOS, watchOS und tvOS – erhalten im Herbst jeweils ein eigenes großes Update, und damit einher geht auch eine Vielzahl neuer Möglichkeiten und Frameworks für Entwickler. Besonders spannend ist dabei ein neues Framework zur Arbeit mit Notifications. Diese waren bisher unter iOS in das UIKit-Framework integriert und wurden über entsprechende Klassen wie *UILocalNotification* oder Systemfunktionen wie *registerUserNotificationSettings(_:)* der Klasse *UIApplication* erstellt und verwendet. Das war zwar einfach, bot aber nicht sonderlich viel Konfigurationsspielraum. Außerdem war es durch die direkte Integration in UIKit nicht immer ersichtlich, welche Funktionen über welche Klasse zur Verfügung stehen.



Das UserNotifications-Framework ist Teil von iOS 10, tvOS 10 und watchOS 3 (Bild 1)

Mit iOS 10 wird sich das nun grundlegend ändern. Zum Umgang mit Notifications führt Apple nämlich dort das neue UserNotifications-Framework ein (Bild 1). Unter diesem Framework werden alle Klassen und Funktionen zusammengefasst, die für die Erstellung und die Arbeit mit Notifications wichtig sind. Darüber hinaus finden sich dort auch einige gänzlich neue Funktionen, mit denen Notifications in iOS ganz neue Möglichkeiten und Optionen zur Verfügung stehen.

Plattformen und Verfügbarkeit

Zunächst etwas zu den Rahmenbedingungen, unter denen das UserNotifications-Framework bereitgestellt wird und eingesetzt werden kann. So kann das Framework für Projekte genutzt werden, die auf iOS 10.0, tvOS 10.0 oder watchOS 3.0 (oder einer jeweils aktuelleren Version dieser Betriebssysteme) basieren. Unter macOS oder älteren OS-Versionen der genannten Systeme kann dieses neue API nicht verwendet werden.

Das ist mitunter für bestehende Projekte problematisch, die bereits auf Notifications setzen. Möchten Sie in diesen die Funktionen und das überarbeitete Notification-API des UserNotifications-Frameworks nutzen, bleiben Ihnen prinzipiell nur zwei Möglichkeiten:

- Entweder aktualisieren Sie Ihre App vollständig auf mindestens iOS 10.0, tvOS 10.0 und watchOS 3.0, um das UserNotifications-Framework problemlos im gesamten Projekt einsetzen zu können, oder
- Sie fahren zweigleisig und splitten Ihren Notification-Code in zwei logisch voneinander getrennte Komponenten; die eine mit der alten Logik, eine andere mit den neuen Funktion des UserNotifications-Frameworks.

Für letztere Variante können Sie idealerweise einen Compiler-Befehl der Programmiersprache Swift verwenden, um die zugrunde liegende Betriebssystemversion zu ermitteln, unter der Ihre App gerade ausgeführt wird. Dabei nutzen Sie die Direktive *#available* und übergeben ihr als Parameter den Namen des Betriebssystems sowie dessen zugehörige Mindestversionsnummer, für die der auszuführende Code funktioniert. Idealerweise verpacken Sie das in eine *if*-Abfrage und führen dann je nach Ergebnis entweder die neuen API-Funktionen aus oder greifen auf die alten zurück. Listing 1 zeigt, wie der Aufbau einer solchen *if*-Abfrage aussehen kann.

Dabei noch ein paar Worte zu dem *#available*-Befehl: Sie können diesem als Parameter die folgenden Werte (jeweils gefolgt von der gewünschten Mindestbetriebssystemversion) übergeben: *iOS, OSX, watchOS, tvOS, iOSApplicationExtension, OSXApplicationExtension, watchOSApplicationExtension* und *tvOSApplicationExtension*.

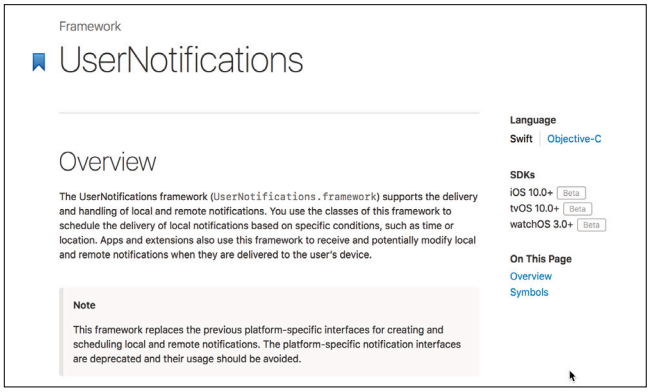
Welche und wie viele dieser Parameter für *#available* Sie dabei übergeben, ist Ihnen überlassen und schlicht davon abhängig, welche Plattformen und Betriebssystemversionen Sie

Listing 1: Aufbau von #available

```
if #available(iOS 10.0, tvOS 10.0, watchOS 3.0, *) {
  // Use new UserNotifications framework...
}
else
{
  // Use previous technologies...
}
```

innerhalb einer Abfrage prüfen müssen. Trennen Sie die gewünschten Parameter mit Versionsnummer einfach nacheinander mittels Komma. Als letzten Parameter setzen Sie in jedem Fall noch einen Stern *, mit dem Apple bereits im Vorhinein mögliche zukünftige Plattformen abdecken möchte.

Sollte es für Sie keine größeren Umstände bedeuten, Ihre App bereits vollumfänglich auf die neuen Betriebssystemversionen von iOS, tvOS und watchOS umzustellen, so würde ich Ihnen dazu raten, das auch zu tun und in Ihrem gesamten Projekt ausschließlich auf das neue UserNotifications-Framework zur Arbeit mit Notifications zurückzugreifen. Nur wenn Sie wirklich nicht darum herumkommen, ältere Betriebssystemversionen zu unterstützen (beziehungsweise das möchten), können Sie *#available* dazu nutzen, Ihren Code passend für beide Notification-Technologien zu strukturieren.



In der Dokumentation des UserNotifications-Frameworks finden sich alle Informationen zu Klassen und Methoden (Bild 2)

Bestandteile des Frameworks

Innerhalb des neuen UserNotifications-Frameworks finden sich alle wichtigen Klassen mitsamt zugehöriger Methoden und Eigenschaften, die für die Arbeit mit Notifications nötig sind (Bild 2). Herzstück ist dabei die Klasse *UNUserNotificationCenter*. Diese ist als Singleton konzipiert; entsprechend wird eine Instanz der Klasse typischerweise mit Hilfe der Klassenmethode *current()* referenziert. Diese liefert die Singleton-Instanz von *UNUserNotificationCenter* für Ihre App zurück.

UNUserNotificationCenter ist deshalb so wichtig, da darüber alle Notifications organisiert werden. Über diese Klasse können Sie den Nutzer über eine Systemanfrage fragen, ob er Notifications für Ihre App erlaubt, Sie können Notification-Kategorien registrieren, in denen Sie spezifische Funktionen für verschiedene Arten von Notifications definieren, sowie Local Notifications terminieren und senden.

Zum Erstellen von Notifications spielen zwei weitere Klassen eine essenzielle Rolle: *UNNotificationRequest* wird dazu verwendet, den Inhalt einer Notification mit einem sogenannten Trigger zu verbinden. Ein Notification-Inhalt wird dabei über die Klasse *UNNotificationContent* abgebildet. Ein solcher Inhalt besitzt beispielsweise unter anderem einen Titel, einen Text oder ein Bild. *UNNotificationTrigger* ist die Grundlage der genannten Trigger. Es handelt sich dabei um eine abstrakte Klasse, von der typischerweise keine Instanzen erstellt werden. Stattdessen bringt das UserNotifications-Framework noch weitere Subklassen von *UNNotificationTrigger* mit, über die Sie festlegen, in welchen Situationen oder zu welcher Zeit eine Local Notification gesendet wird.

Das, was *UNNotificationRequest* für das Versenden und Planen von Notifications ist, ist *UNNotificationResponse* für das Eingehen und dazugehörige Auswerten einer Notification. Ein solches Objekt erhalten wir, wenn eine Notification im System eingeht und von unserer App ausgewertet werden muss (beispielsweise aufgrund einer Action, die der Nutzer im Kontext der Notification ausführt). Dabei enthält ein Objekt der Klasse *UNNotificationResponse* einen Verweis auf ein UNNotification-Objekt, das wiederum den ursprünglichen UNNotificationRequest enthält. Bei UNNotification handelt es sich somit um ein Objekt, das eine eingehende Notification abbildet, und nicht – wie der Name es womöglich vermuten lassen würde – um den Inhalt einer Notification; für Letzteres ist die bereits vorgestellte Klasse *UNNotificationContent* verantwortlich.

Damit kennen Sie nun bereits einmal einige der wichtigsten und grundlegendsten Klassen des UserNotifications-Frameworks. In den folgenden Abschnitten werde ich detailliert auf deren jeweilige Funktionsweise eingehen sowie weitere Klassen und Bestandteile des Frameworks vorstellen. Dabei gehe ich in der typischen Reihenfolge vor, die durchlaufen werden muss, um zunächst Notifications zu erstellen und zu versenden sowie anschließend empfangen und auswerten zu können.

Autorisierung erfragen

Um überhaupt einem Nutzer eine Notification (egal ob Local oder Remote) schicken zu können, müssen Sie aus Ihrer App heraus zunächst einmal eine Autorisierung dafür erfragen. In dieser Autorisierung legen Sie fest, welche Art von Notifications Ihre App senden soll, woraufhin ein vom jeweiligen Betriebssystem automatisch erzeugter Alert den Nutzer auf diesen Umstand hinweist. Er kann darüber dann der Verwendung von Notifications für Ihre App zustimmen oder diese ablehnen. Im letzteren Fall wird der Nutzer niemals Notifications Ihrer App erhalten, selbst wenn Sie welche senden; das zugrunde liegende System wird sie dann einfach nicht zustellen.

Aktuell gibt es insgesamt vier verschiedene Arten von Notifications, die Sie verwenden können. Dabei spielt es keine Rolle, welche und wie viele Sie davon nutzen möchten. Wichtig ist nur, dass Sie beim Erfragen der Nutzerautorisierung alle Arten angeben, die für Sie relevant sind. Diese Arten sind dabei in der Structure *UNAuthorizationOptions* des UserNotifications-Frameworks definiert, es handelt sich dabei um die Folgenden:

- *badge:* Mit dieser Form der Notification können Sie den Wert des Badge-Icons verändern, das als roter Kreis im rechten oberen Bereich des zugehörigen App-Icons einer App angezeigt wird.
- *sound:* Erlaubt es Ihnen, Sound bei Eingang einer Notification wiederzugeben.
- *alert:* Sicherlich einer der am häufigsten verwendeten Typen von Notifications. Wie der Name bereits sagt, wer- ▶

den solche Notifications als Alert auf dem Display angezeigt und können darüber auch die meisten Informationen transportieren (wie Texte und Bilder).
- *carPlay:* Sollen Notifications auch innerhalb einer CarPlay-Umgebung angezeigt werden, wird dieser Typ ebenfalls noch benötigt.

Um nun die gewünschten Arten von Notifications für Ihre App zu registrieren, müssen Sie die Methode *requestAuthorization(options:completionHandler:)* des Singletons der Klasse *UNUserNotificationCenter* aufrufen. Diese Methode erhält dabei zwei Parameter. Der erste erwartet die gewünschten Notification-Arten, die Sie verwenden möchten. Diese übergeben Sie dabei einfach in Form eines Arrays und fügen dort alle Werte von *UNAuthorizationOptions* ein, die für Sie relevant sind.

Der zweite Parameter ist ein Closure. Dieses Closure wird vom System aufgerufen, sobald der Nutzer den Autorisierungsprozess abgeschlossen hat. Durch zwei Parameter, die Teil des Closures sind und die Ihnen somit vom System zur Verfügung gestellt werden, können Sie einerseits auslesen, ob der Nutzer der Verwendung von Notifications für Ihre App zugestimmt hat oder nicht, und andererseits sehen, ob möglicherweise ein Fehler aufgetreten ist. Diese Information können Sie beispielsweise dazu nutzen, im Fall einer Ablehnung von Notifications durch den Nutzer gar nicht erst Notifications an ihn zu versenden (tun Sie es dennoch, werden diese, wie bereits beschrieben, einfach nicht zugestellt).

Sobald Sie diese Methode in Ihrem Code aufrufen, erscheint die vom System bereitgestellte Meldung, in der der Nutzer gefragt wird, ob dieser die Verwendung von Notifications für Ihre App erlaubt. Diese Meldung erscheint dabei nur einmalig; sobald der Nutzer diese das erste Mal entweder positiv oder negativ bestätigt hat, wird diese Einstellung im System gespeichert und der Aufruf der Methode *requestAuthorization(options:completionHandler:)* führt zu keinen erneuten Autorisierungsmeldungen. Entsprechend können Sie diesen Befehl durchaus auch dann noch aufrufen, wenn der Nutzer bereits seine Entscheidung bezüglich der Autorisierung gefällt hat; es erscheint dann schlicht und einfach keine entsprechende Anfrage mehr.

Ein guter Ort, um diese Methode aufzurufen, ist innerhalb einer iOS-App somit beispielsweise Ihre *UIApplicationDelegate*-konforme *AppDelegate*-Klasse und die Methode *application(_:didFinishLaunchingWithOptions:)*.

Listing 2 zeigt einmal beispielhaft die Autorisierungsanfrage für Alert- und Sound-Notifications. Dabei wird auch mit Hilfe des *completionHandler*-Parameters der Methode *requestAuthorization(options:completionHandler:)* überprüft, ob der Nutzer die Verwendung von Notifications autorisiert hat oder nicht.

Listing 2: Autorisierungsanfrage

```
import UIKit
import UserNotifications

class AppDelegate: NSObject, UIApplicationDelegate
{
  private func application(_ application:
  UIApplication, didFinishLaunchingWithOptions
  launchOptions: [NSObject : AnyObject]? = [:]) ->
  Bool {
    let userNotificationCenter =
    UNUserNotificationCenter.current()
    let authorizationOptions:
    UNAuthorizationOptions = [.alert, .sound]
    userNotificationCenter.requestAuthorization
    (options: authorizationOptions) { (granted:
    Bool, error: Error?) in
      if granted {
        print("Nutzer hat der Verwendung von
        Notifications zugestimmt.")
      } else {
        print("Nutzer hat die Verwendung von
        Notifications nicht erlaubt.")
      }
    }
    return true
  }
}
```

Inhalt einer Notification erstellen

Wurde die Autorisierung zum Versenden von Notifications erfolgreich beim Nutzer erfragt und genehmigt, haben Sie damit die Möglichkeit, nun auch entsprechend Notifications aus Ihrer App heraus zu versenden und dem Nutzer anzuzeigen.

Damit Sie aber überhaupt eine Notification zu einem bestimmten Zeitpunkt oder einem Ereignis auch tatsächlich versenden und anzeigen können, brauchen Sie zunächst einmal einen Inhalt für Ihre Notfication. Dieser Inhalt setzt sich aus Informationen wie dem Titel oder einem Text sowie einem möglichen Sound oder einem geänderten Badge-Wert des zugehörigen App-Icons zusammen. Dabei können Sie beim Inhalt nur jene Elemente verwenden, die Sie zuvor auch mittels der Methode *requestAuthorization(options:completionHandler:)* erfragt haben. Haben Sie sich beispielsweise nicht für die Option *sound* entschieden, können Sie auch keinen Sound über den Inhalt Ihrer Notification wiedergeben, selbst wenn Sie einen solchen konfigurieren.

Herzstück dieses Inhalts einer Notification ist die Klasse *UNNotificationContent*. Diese verfügt über verschiedene Properties, um auf die Inhalte einer Notification zugreifen zu können. Darunter gehören solche wie *title, subtitle, body, badge* oder *sound*. Die erstgenannten drei können Sie dann verwenden, wenn Sie als Option für Ihre Notifications bei der Autorisierung den Wert *alert* übergeben haben; *badge* können Sie mit Hilfe der Option *badge* und *sound* – wie eben bereits beschrieben – mit der Option *sound* verwenden. Ist eine Option nicht autorisiert, können Sie zwar dennoch einen Wert für das entsprechende Element setzen, allerdings spielt dieser letzten Endes keine Rolle.

Listing 3: UNMutableNotificationContent

```
let notificationContent =
UNMutableNotificationContent()
notificationContent.title = "My notification"
notificationContent.subtitle =
"My notification subtitle"
notificationContent.body = "My notification body"
notificationContent.badge = 3
notificationContent.sound =
UNNotificationSound(named: "MySound.aiff")
notificationContent.launchImageName =
"NotificationLaunchImage"
```

Die Klasse *UNNotificationContent* enthält also den Inhalt einer Notification. Die Klasse hat nur ein Problem: Alle genannten Eigenschaften für Notifications setzen sich aus Read-only-Properties zusammen, sie können also nur gelesen und nicht geschrieben werden. Beim Erstellen einer neuen Notification mitsamt gewünschtem Inhalt ist dieses Verhalten natürlich vollends kontraproduktiv; schließlich geht es da ja gerade darum, einen neuen Inhalt für eine Notification zu erstellen.

Zu diesem Zweck verfügt das UserNotifications-Framework über eine zusätzliche Subklasse von *UNNotificationContent* mit dem Namen *UNMutableNotificationContent*. Diese bringt ebenfalls alle die aus *UNNotificationContent* bekannten Properties mit, allerdings mit dem großen Unterschied, dass diese Properties nun auch schreibbar sind und ihnen somit passende Werte zugewiesen werden können.

Die Klasse *UNMutableNotificationContent* verwenden wir somit immer dann, wenn wir neue Inhalte für eine Notification erstellen möchten, während *UNNotificationContent* zum reinen Auslesen der Informationen einer Notification dient. Objekte letzterer Klasse werden wir bei weiteren Funktionen des UserNotifications-Frameworks noch des Öfteren zu Gesicht bekommen, da die unveränderlichen UNNotificationContent-Objekte häufig vom System als Parameter genutzt werden, um auf eine Notification hinzuweisen. Schließlich ist es bei einer eingegangenen Notification zunächst einmal per se nicht sinnvoll, deren Inhalt noch zu verändern, und das wird durch Verwendung von Objekten der Klasse *UNNotificationContent* somit auch explizit ausgeschlossen.

Listing 3 zeigt nun, wie sich ein neuer Inhalt für eine Notification mit Hilfe eines UNMutableNotificationContent-Objekts erstellen und erzeugen lässt. Wie wir diesen Inhalt einer Notification dann in einem nächsten Schritt dazu verwenden, um eine passende und entsprechende Notification zu senden, erfahren Sie gleich.

Notification-Trigger

Wir haben nun also eine konfigurierte Nachricht für eine Notification. Bevor diese gesendet wird, muss aber noch eine essenzielle Frage beantwortet werden: Wann beziehungsweise zu welchem Ereignis soll die Notification denn dem Nutzer angezeigt werden? Was muss passieren, damit der Nutzer diese Notification erhält? Gibt es dafür ein festgelegtes Datum mitsamt Uhrzeit? Oder einen Ort, an dem die Notification erscheinen soll?

Listing 4: UNCalendarNotificationTrigger-Objekt

```
var dateComponents = DateComponents()
dateComponents.hour = 8
dateComponents.minute = 30
let calendarNotificationTrigger =
UNCalendarNotificationTrigger(dateMatching:
dateComponents, repeats: true)
```

Diese Fragen gilt es zu klären, und das UserNotifications-Framework bringt eine eigene abstrakte Klasse mit, um genau diese Problematik abzubilden: *UNNotificationTrigger*. Mit Hilfe von Subklassen dieser Klasse ist es möglich, Bedingungen festzulegen, zu denen eine Notification beim Nutzer ankommen soll. Wie bereits beschrieben, ist die Klasse *UNNotificationTrigger* selbst abstrakt und bringt lediglich grundlegende Informationen und Eigenschaften für mögliche Notification-Trigger mit. Sie werden nie Instanzen dieser Klasse selbst erstellen und als Trigger verwenden. Stattdessen verwenden Sie eine der verfügbaren *UNNotificationTrigger*-Subklassen, die jeweils andere Voraussetzungen zum Senden von Notifications zur Verfügung stellen. Je nachdem, unter welchen Bedingungen und zu welchen dieser Voraussetzungen Sie eine Notification senden möchten, verwenden Sie die dazu passende Klasse als Trigger für Ihre Notification. Im Folgenden stelle ich Ihnen einmal die verfügbaren *UNNotificationTrigger*-Subklassen kurz vor.

Betrachten wir dabei zunächst einmal die Klasse *UNCalendarNotificationTrigger*. Diesen Trigger verwenden Sie, wenn Sie eine Notification zu einem festen Zeitpunkt, bestehend aus Datum und Uhrzeit, feuern möchten. Dazu erstellen Sie Objekte dieser Klasse mit Hilfe des zugehörigen Initializers *init(dateMatching:repeats:)*. Der erste Parameter davon ist *dateComponents* und ist vom Typ *DateComponents*. Dabei handelt es sich um eine Structure, die es Ihnen erlaubt, ein Datum und eine Uhrzeit anhand verschiedenster Eigenschaften abzubilden. So gibt es beispielsweise Properties wie *hour*, *minute* oder *month*, mit denen Sie eben genau diese Informationen für einen spezifischen Zeitpunkt abbilden können.

Listing 4 zeigt einmal beispielhaft, wie ein Objekt vom Typ *DateComponents*, das der Uhrzeit *08:30* (unabhängig vom Datum) entspricht, erstellt und anschließend mit Hilfe des Initializers der Klasse *UNCalendarNotificationTrigger* ein ebensolcher Trigger erzeugt wird. Interessant ist dabei übrigens auch noch der zweite Parameter *repeats* des Initializers *init(dateMatching:repeats:)*. Ist dieser auf *true* gesetzt, dann wird die geplante Notification zum gegebenen Zeitpunkt wiederholt, selbst wenn sie bereits einmal gefeuert wurde. Im Beispiel bedeutet das, dass eine Notification mit dem erstellten calendarNotificationTrigger-Objekt jeden Tag um 08:30 Uhr morgens erneut gesendet wird. ▶

Listing 5: UNLocationNotificationTrigger-Objekt

```
let coordinates =
CLLocationCoordinate2DMake(49.96778343788903,
9.174244946895568)
let circularRegion = CLCircularRegion.init(center:
coordinates, radius: 2000.0, identifier: "Home")
circularRegion.notifyOnEntry = true
circularRegion.notifyOnExit = false
let locationNotificationTrigger =
UNLocationNotificationTrigger.init(region:
circularRegion, repeats: false)
```

Der nächste Trigger wird mit Hilfe der Klasse *UNLocationNotificationTrigger* abgebildet. Darüber ist es möglich, eine Notification an einem spezifischen Ort zu senden. Dieser Ort wird dabei in Form eines CLRegion-Objekts abgebildet. CLRegion ist eine abstrakte Klasse aus dem CoreLocation-Framework von Apple. Innerhalb des Frameworks finden sich verschiedene Subklassen zum Abbilden eines spezifischen Ortes, auf deren detaillierte Ausführung ich an dieser Stelle aber verzichten werde; immerhin geht es hier um das UserNotifications- und nicht das CoreLocation-Framework.

Um einen Trigger der Klasse *UNLocationNotificationTrigger* zu erstellen, verwendet man den Initializer *init(region: repeats:)*. Der erste Parameter *region* entspricht dem beschriebenen CLRegion-Objekt, während der zweite Parameter *repeats* wieder angibt, ob die Notification nach dem erstmaligen Senden erneut bei Erreichen derselben Location gesendet werden soll oder nicht.

Listing 5 zeigt einmal ein vollständiges Beispiel zum Erstellen eines CLRegion-Objekts sowie darauf aufbauend eines UNLocationNotificationTriggers.

Betrachten wir nun Trigger Numero drei: *UNTimeIntervalNotificationTrigger*. Diese Klasse dient dazu, Trigger abzubilden, die nach einer festgelegten Dauer gefeuert werden. Neue Instanzen dieses Triggers werden mit Hilfe des Initializers *init(timeInterval:repeats:)* erzeugt, wobei der erste Parameter *timeInterval* einen Zahlenwert in Sekunden erhält, nach deren Ablauf eine mit diesem Trigger erstellte Notification gefeuert wird. Wundern Sie sich dabei nicht über den Typ dieses Parameters, der als *TimeInterval* spezifiziert ist; es handelt sich dabei um nichts weiter als einen Type-Alias für den Typ *Double*.

Wie sich ein einfacher Trigger vom Typ *UNTimeIntervalNotificationTrigger* erstellen lässt, der nach einer Stunde ausgelöst wird, zeigt ein kleines Beispiel in Listing 6.

Bleibt zu guter Letzt noch der Trigger vom Typ *UNPushNotificationTrigger*. Dabei handelt es sich um eine Sonderform, denn Instanzen dieser Klasse erstellen Sie typischerweise nicht. Vielmehr wird dieser Trigger automatisch all jenen Notifications zugewiesen, die als Push-Notification an den Nutzer gesendet werden. Das geschieht automatisch und Sie müssen dafür beim Konfigurieren von Push-Notifications nichts Zusätzliches beachten. Vielmehr können Sie bei Erhalt einer Notification in Ihrer App anhand dieses Triggers ermitteln, ob es sich bei der erhaltenen Notification um eine Push- oder eben doch – im Fall eines anderen Triggers – um eine Local Notification handelt.

Listing 6: UNTimeIntervalNotificationTrigger-Objekt

```
let oneHourInSeconds: TimeInterval = 60 * 60
let timeIntervalNotificationTrigger =
UNTimeIntervalNotificationTrigger(timeInterval:
oneHourInSeconds, repeats: true)
```

Erstellen eines Notification-Requests

Wir haben nun also zwei essenzielle Bestandteile einer jeden Notification kennengelernt: den eigentlichen Inhalt einer Notification in Form eines UNNotificationContent-Objekts sowie das Festlegen des Auslöseereignisses einer Notification mittels Instanz einer passenden Subklasse von *UNNotificationTrigger*. Im nächsten Schritt geht es darum, diese beiden Elemente zusammenzuführen und daraus einen sogenannten Notification-Request zu generieren. Genau ein solcher Request ist es dann, den wir im Singleton des UNUserNotificationCenters hinzufügen, um eine Notification im System zu registrieren und zu feuern, sobald der gesetzte Trigger erfüllt ist.

Ein solcher Request wird mit Hilfe der Klasse *UNNotificationRequest* erstellt. Objekte dieser Klasse setzen sich aus den folgenden drei Bestandteilen zusammen, die als gleichnamige Read-only-Properties auch ausgelesen werden können:

- *identifier:* Ein eindeutiger Identifier, der zur Identifizierung eines Notification-Requests dient. Den Namen dieses Identifiers können Sie selbst frei festlegen.
- *content:* Hierbei handelt es sich um den Inhalt Ihrer Notification in Form eines UNNotificationContent-Objekts.
- *trigger:* Das Ereignis, zu dem die Notification gesendet werden soll. Es handelt sich dabei um ein Objekt einer der vorgestellten passenden Subklassen von *UNNotificationTrigger*.

Interessantes Detail dabei: Bei der letztgenannten Property *trigger* handelt es sich um ein Optional, sie kann somit also auch *nil* entsprechen. Tatsächlich muss eine Notification über keinen Trigger verfügen. Fehlt ein solcher Trigger für eine Notification, wird diese einfach direkt und umgehend gefeuert, sobald Sie einen Notification-Request über das Singleton des UNUserNotificationCenters registrieren.

Um eine neue Instanz der Klasse *UNNotificationRequest* zu erstellen, müssen Sie den zugehörigen Initializer *init(identifier:content:trigger:)* aufrufen und verwenden. Dabei werden für alle drei vorgestellten Properties die jeweils zugehörigen Werte von Ihnen wie gewünscht gesetzt und übergeben. In einem letzten Schritt kann dann eine solche *UNNotificationRequest*-Instanz mit Hilfe der Methode *add(_:withCompletionHandler:)* im System registriert werden.

Listing 7 zeigt einmal den kompletten Ablauf dieses Vorgehens mit Hilfe der vorangegangenen Beispiele, angefangen beim Erstellen eines UNNotificationContents und eines Triggers über das Initialisieren eines Notification-Requests (der Content und Trigger zusammenfasst) bis hin zum finalen Registrieren dieses Requests im System mit Hilfe des Singletons des UNUserNotificationCenters.

Dabei möchte ich abschließend noch ein paar kurze Worte zum letzten Parameter *completionHandler* der Methode *add(_:withCompletionHandler:)* verlieren. Es handelt sich dabei um ein Closure, dem als Parameter ein optionales Error-Objekt vom System übergeben wird. Das Closure wird aufgerufen, sobald das System den Vorgang zum Hinzufügen Ihres Notification-Requests abgeschlossen hat. Sie können es dazu verwenden, auf etwaige Fehler oder Probleme zu reagieren, die möglicherweise bei diesem Hinzufügen des Notification-Requests auftreten.

Der Parameter *completionHandler* selbst ist ein Optional, er wird also nur gesetzt sein, wenn es auch tatsächlich zu einem Problem gekommen ist. In diesen Fällen aber können Sie mit Hilfe des übergebenen Error-Objekts ermitteln, was möglicherweise schiefgelaufen ist, und entsprechend weitere Aktionen durchführen.

Listing 7: Erstellen eines Notification-Requests

```
// 1. Create notification content
let notificationContent =
UNMutableNotificationContent()
notificationContent.title = "My notification"
notificationContent.subtitle = "My notification
subtitle"
notificationContent.body = "My notification body"
notificationContent.badge = 3
notificationContent.sound =
UNNotificationSound(named: "MySound.aiff")
notificationContent.launchImageName =
"NotificationLaunchImage"
// 2. Create notification trigger
var dateComponents = DateComponents()
dateComponents.hour = 8
dateComponents.minute = 30
let calendarNotificationTrigger =
UNCalendarNotificationTrigger(dateMatching:
dateComponents, repeats: true)

// 3. Create notification request
let notificationRequest = UNNotificationRequest
(identifier: "MyNotification", content:
notificationContent, trigger:
timeIntervalNotificationTrigger)

// 4. Register notification request
UNUserNotificationCenter.current().
add(notificationRequest, withCompletionHandler: nil)
```

Mit dem bisher gezeigten und vorgestellten Verfahren können Sie Local Notifications (also lokal auf dem jeweiligen Endgerät erstellte und im System registrierte Notifications) erstellen und für den Versand mittels passendem Trigger vorbereiten. Ist die Bedingung des zugrunde liegenden Triggers erfüllt, erscheint entsprechend die jeweilige Notification auf dem Gerät des Nutzers.

Allerdings kann der Nutzer mit derartigen Notifications keine wirklichen Aktionen durchführen. Mehr, als sie zu betrachten und wieder auszublenden, ist nicht möglich. In manchen Fällen ist es aber durchaus sinnvoll, den eigenen Notifications direkt passende Aktionen mit auf den Weg zu geben, die der Nutzer dann darauf ausführen kann. Nehmen wir beispielsweise eine Erinnerung für das Fertigstellen einer Aufgabe aus einer Aufgaben-App: Hier könnte es sinnvoll sein, dem Nutzer eine Aktion *Erledigt* anzubieten, mit der er die präsentierte Aufgabe bereits als erfüllt markieren kann.

Das UserNotifications-Framework bietet genau eine solche Lösung für die genannte Problematik und verwendet dazu Kategorien und Aktionen. Eine Aktion entspricht dabei einem spezifischen Befehl wie dem beispielhaft genannten *Erledigt* zum Markieren einer Aufgabe als erfüllt. Zusammengehörige Aktionen werden in einer Kategorie zusammengefasst, und genau eine solche Kategorie ist es, die einer Notification zugewiesen werden kann, um dieser Notification die in der Kategorie enthaltenen Aktionen zuzuweisen.

All diese Elemente betrachten wir nun einmal im Detail. Beginnen wir dabei mit den Aktionen.

Aktionen werden mittels Instanzen der Klasse *UNNotificationAction* abgebildet. Die Klasse verfügt dabei über die folgenden drei Eigenschaften:

- *identifier:* Ein eindeutiger, von Ihnen festgelegter Identifier für eine spezifische Action.
- *title:* Der Titel der Action.
- *options:* Mögliche zusätzliche Bedingungen und Optionen, die mit der Action in Zusammenhang stehen.

Alle diese Werte werden – ähnlich wie bei einem UNNotificationRequest – mit Hilfe des zugehörigen Initializers *init (identifier:title:options:)* gesetzt. Dabei möchte ich zu diesem Zeitpunkt den letzten Parameter *options* ein wenig erläutern. Dieser ist vom Typ *UNNotificationActionOptions* und stellt ein sogenanntes OptionSet dar. Er verfügt somit über verschiedene festgelegte Eigenschaften, von der keine, eine oder mehrere ausgewählt und gesetzt werden können. Bevor wir betrachten, wie ein Wert für diesen *options*-Parameter konkret gesetzt werden kann, möchte ich die zur Verfügung stehenden Optionen der Structure *UNNotificationActionOptions* vorstellen:

- *authenticationRequired:* Ist diese Option gesetzt, kann die entsprechende Aktion nur ausgeführt werden, wenn der Nutzer sein Gerät entsperrt hat. Ist das nicht der Fall, wird das System ihn automatisch dazu auffordern, eine Entsperrung des entsprechenden Geräts vorzunehmen, bevor er die zugrunde liegende Aktion durchführen kann.
- *destructive:* Diese Option sorgt für eine optische Veränderung der Schaltfläche der entsprechenden Action. Sie ▶

soll dazu dienen, den Nutzer auf einen Vorgang hinzuweisen, der möglicherweise zum Löschen von Daten führt (beispielsweise das Entfernen einer Aufgabe aus einer Aufgaben-App). Funktionale Eigenschaften bringt diese Option nicht mit sich, sie dient rein der optischen Hervorhebung der zugehörigen Action.

- *foreground:* Ist diese Option gesetzt, wird bei Ausführung der entsprechenden Action die zugehörige App im Vordergrund gestartet und ausgeführt.

Wie beschrieben können Sie für eine Action keine, eine oder mehrere dieser Optionen setzen. Dabei übergeben Sie ein Array mit den gewünschten Optionen (oder ein leeres Array, wenn Sie keine der Optionen für Ihre Action verwenden möchten) als Wert für den *options*-Parameter.

Listing 8 zeigt Ihnen die beispielhafte Erstellung einer *Erledigt*-Action für eine Aufgaben-App, der beispielhaft die Option *authenticationRequired* zugewiesen wird.

Auf diese gezeigte Art und Weise können Sie beliebige Actions für Ihre Notifications erstellen. Um eine oder mehrere davon Ihren Notifications zuweisen zu können, müssen Sie diese aber noch in passenden Kategorien zusammenfassen.

Erstellen von Notification-Categories

Um Actions in Kategorien zusammenzufassen, werden Objekte der Klasse *UNNotificationCategory* des UserNotifications-Frameworks verwendet. Objekte dieser Klasse verfügen wieder über einen eindeutigen Identifier, über den die Kategorie in Ihrem Projekt eindeutig identifiziert werden kann, sowie ein Array an UNNotificationAction-Objekten, in denen all jene Actions enthalten sind, über die die zu erstellende Kategorie verfügen soll. Darüber hinaus können Sie auch für Objekte dieser Klasse zusätzliche Optionen setzen, die vom Typ *UNNotificationCategoryOptions* sind. Auch hierbei handelt es sich – wie schon bei *UNNotificationActionOptions* bei den Actions – um ein OptionSet, sprich Sie können entweder keine, eine oder beliebig viele der zur Verfügung stehenden Optionen setzen und in Form eines Arrays übergeben.

Um all diese Eigenschaften einer *UNNotificationCategory* festzulegen, bringt auch diese Klasse einen eigenen Initializer namens *init(identifier:actions:intentIdentifiers:options:)* mit. UNNotificationCategory-Objekte werden ausschließlich über diesen Designated Initializer erstellt und erzeugt. Dabei möchte ich die vier verschiedenen Parameter dieses Initializers an dieser Stelle kurz erläutern:

- *identifier:* Hierbei handelt es sich um einen eindeutigen Bezeichner, mit dem Sie die zugehörige Kategorie in Ihrem Projekt identifizieren können.
- *actions:* Dieser Parameter erwartet ein Array all der Actions in Form von UNNotificationAction-Objekten, die innerhalb dieser Kategorie verwendet werden sollen.
- *intentIdentifiers:* Hierbei handelt es sich um Strings, die Sie im Zusammenspiel mit Notifications dieser Kategorie bei der Verwendung von Siri nutzen möchten. Da wir uns hier in die Funktionen und Eigenschaften des Intents-Frameworks und die Nutzung von Siri in eigenen Apps begeben, spare ich weitere Erläuterungen für diesen Artikel aus, da das ansonsten den Rahmen sprengen würde.
- *options:* Hier legen Sie zusätzliche Optionen fest, die für diese Kategorie gültig sind.

Listing 8: UNNotificationAction-Objekt

```
let doneAction = UNNotificationAction(identifier:
"Done", title: "Erledigt",
options: [.authenticationRequired])
```

Listing 9: UNNotificationCategory-Objekt

```
let notificationCategory = UNNotificationCategory
(identifier: "MyCategory", actions: [doneAction],
intentIdentifiers: [], options: [.allowInCarPlay])
notificationContent.categoryIdentifier =
"MyCategory"
```

Auch hier möchte ich wieder auf den letzten Parameter *options* noch einmal im Detail eingehen. Ich schrieb bereits, dass es sich dabei um ein OptionSet der Structure *UNNotificationCategoryOptions* handelt. Folgende Optionen können Sie über diesen Typ für Ihre Notification-Category setzen:

- *customDismissAction:* Erlaubt das Senden angepasster Aktionen beim Ausblenden der Notification über den Delegate des UNUserNotificationCenters.
- *allowInCarPlay:* Erlaubt das Anzeigen der zugehörigen Notifications auf einem CarPlay-fähigen System.

Mit diesen Informationen lassen sich nun Instanzen der Klasse *UNNotificationCategory* erstellen. Damit eine Notification die Actions einer solchen Notification-Category besitzt, müssen Sie beim Erstellen eines Notification-Inhalts den Identifier dieser Kategorie der Property *categoryIdentifier* der Klasse *UNMutableNotificationContent* zuweisen. Listing 9 zeigt ein einfaches Beispiel, in dem eine neue Notification-Category erstellt und dieser die *allowInCarPlay*-Option sowie die im vorigen Beispiel erzeugte Action zugewiesen wird. Anschließend wird der Identifier dieser erstellte Notification-Category der Property *categoryIdentifier* des UNMutableNotificationContent-Objekts zugewiesen, damit diese Notification über die entsprechenden Actions der Kategorie verfügt.

Notification-Actions abfangen

Haben Sie einer Notification eine Kategorie mit Actions zugewiesen, möchten Sie natürlich auch auf eine durch den Nutzer ausgelöste Action in Ihrem Code reagieren können. Zu diesem Zweck benötigen Sie ein Delegate-Objekt, das konform zum UNUserNotificationCenterDelegate-Protokoll ist und das der Property *delegate* des Singletons des UNUserNotificationCenter-Objekts zugewiesen ist. Wichtig ist dabei, dass die Zuweisung eines entsprechenden Delegate-Objekts

zu UNUserNotificationCenter spätestens mit dem Start der App erfolgt.

Die Klasse dieses Objekts muss sodann die optionale Methode *userNotificationCenter(_:didReceive:withCompletionHandler:)* des UNUserNotificationCenterDelegate-Protokolls implementieren, um durch Notifications ausgelöste Actions auswerten zu können. Der wichtigste Parameter dieser Methode ist dabei der zweite mit Namen *response*. Dieses Objekt ist vom Typ *UNNotificationResponse* und enthält alle Informationen zu der Notification sowie der Action, die über diese Notification ausgelöst wurde. Dazu bringt diese Klasse zwei Properties mit, um die entsprechenden Informationen auszuwerten:

- *actionIdentifier:* Hierbei handelt es sich um den von Ihnen festgelegten eindeutigen Identifier des UNNotificationAction-Objekts, dessen Action der Nutzer über die zugehörige Notification ausgewählt hat.
- *notification:* Diese Property gibt Ihnen Zugriff auf ein UNNotification-Objekt. Dieses Objekt gewährt Ihnen Zugriff auf das Datum, zu dem die Notification übermittelt wurde, sowie den zugrunde liegenden UNNotificationRequest, der für die Notification im UNUserNotificationCenter registriert wurde.

Mit diesen Informationen haben Sie alles, was Sie brauchen, um zu erfahren, welche Notification mit welcher Action der Nutzer ausgelöst hat. Sie können diese dazu nutzen, die gewünschte Aktion in Ihrem Code auszuführen.

Anschließend ist es wichtig, den letzten Parameter der Delegate-Methode *userNotificationCenter(_:didReceive:withCompletionHandler:)* namens *completionHandler* aufzurufen. Es handelt sich dabei um ein Closure ohne Parameter und ohne Rückgabewert. Der Aufruf dieses Closures dient lediglich dazu, dem System mitzuteilen, dass Sie die zugrunde liegende Aktion der eingegangenen Notification verarbeitet haben. Sie sollten ihn so schnell wie nur möglich aufrufen.

Listing 10 zeigt beispielhaft eine Klasse namens *UserNotificationCenterDelegate*, die konform zum UNUserNotificationCenterDelegate-Protokoll ist und die genannte Methode *userNotificationCenter(_:didReceive:withCompletionHandler:)* implementiert. Ebenso zeigt es die Erweiterung der Klasse *AppDelegate* um eine neue Property namens *userNotificationCenterDelegate*, die einen Verweis auf die Klasse *UserNotificationCenterDelegate* hält und diesen in der *UIApplicationDelegate*-Methode *applicationDidFinishLaunching(_:)* setzt.

Notifications im Vordergrund abfangen

Das eben vorgestellte UNUserNotificationCenterDelegate-Protokoll bringt noch eine weitere Methode mit. Diese dient dazu, Notifications abzufangen und eine mögliche angepasste Aktion auszuführen, wenn sich während des Eintreffens der Notification die zugehörige App gerade im Vorder- ▶

Listing 10: Reaktion auf Actions einer Notification

```
class AppDelegate: NSObject, UIApplicationDelegate {
  private let userNotificationCenterDelegate =
  UserNotificationCenterDelegate()
  func applicationDidFinishLaunching
  (_ application: UIApplication) {
    UNUserNotificationCenter.current().delegate
    = userNotificationCenterDelegate
  }
}
class UserNotificationCenterDelegate: NSObject,
UNUserNotificationCenterDelegate {
  private func userNotificationCenter(_ center:
  UNUserNotificationCenter, didReceive response:
  UNNotificationResponse, withCompletionHandler
  completionHandler: () -> Void) {
    print("Notification action: \
    (response.actionIdentifier)")
    print("Notification request:
    \(response.notification.request)")
    completionHandler()
  }
}
```

grund befindet. Sie hört auf den Namen *userNotificationCenter(_:willPresent:withCompletionHandler:)* und ist ähnlich aufgebaut wie die zuvor vorgestellte Protokollmethode *userNotificationCenter(_:didReceive:withCompletionHandler:)*. Ihr zweiter Parameter *notification* ist vom Typ *UNNotification* und enthält somit – wie bereits gesehen – einen Verweis auf die eingegangene Notification mit allen zugehörigen Informationen. Diese können Sie dazu nutzen, Ihre App entsprechend zu aktualisieren.

Der letzte Parameter dieser Protokollmethode namens *completionHandler* ist wieder ein Closure und muss auf jeden Fall innerhalb Ihrer Implementierung von *userNotificationCenter(_:willPresent:withCompletionHandler:)* aufgerufen werden.

Listing 11: Reaktion auf Eingang einer Notification

```
class UserNotificationCenterDelegate: NSObject,
UNUserNotificationCenterDelegate {
  private func userNotificationCenter(_ center:
  UNUserNotificationCenter, willPresent notification:
  UNNotification, withCompletionHandler completion
  Handler: (UNNotificationPresentationOptions) ->
  Void) {
    print("Notification request:
    \(notification.request)")
    completionHandler([.badge, .alert])
  }
}
```

Dabei erwartet dieses Closure einen Parameter vom Typ *UNNotificationPresentationOptions*. Es handelt sich dabei um ein OptionSet, über das Sie definieren, ob und in welcher Form Sie die eingegangene Notification dem Nutzer noch zusätzlich anzeigen möchten. Dabei bietet Ihnen dieser Typ die folgenden Optionen:

- *badge:* Aktualisiert den Badge-Value des App-Icons entsprechend der eingegangenen Notification.
- *sound:* Spielt den Sound der entsprechend eingegangenen Notification.
- *alert:* Zeigt den Alert der entsprechend eingegangenen Notification.

Je nachdem, ob und welche Informationen Sie dem Nutzer zu Ihrer Notification noch anzeigen wollen, während Ihre App im Vordergrund ist, übergeben Sie keinen, einen oder mehrere dieser Parameter in Form eines Arrays dem *completionHandler*-Closure als Parameter. Listing 11 zeigt ein entsprechendes Beispiel. Dabei wird die Klasse *UserNotificationCenterDelegate* um die Protokollmethode *userNotificationCenter(_:willPresent:withCompletionHandler:)* erweitert und dabei der Badge-Value entsprechend der Notification angepasst sowie der zugehörige Alert eingeblendet.

Weitere Funktionen

Bis zu diesem Punkt haben Sie nun alle wichtigen Schritte, Abläufe und Klasse kennengelernt, die im Zusammenspiel mit der Nutzung von Notifications über das neue UserNotifications-Framework wichtig sind. Nun möchte ich Ihnen an dieser Stelle noch ein paar weitere Funktionen der *UNUserNotificationCenter*-Klasse vorstellen, mit denen Sie Ihre Notifications verwalten und bearbeiten können. Die Deklaration all dieser von mir im Folgenden vorgestellten Methoden sehen Sie in Listing 12.

Betrachten wir die gezeigten Methoden einmal der Reihe nach: Mit Hilfe von *getNotificationSettings(completionHandler:)* erhalten Sie ein Objekt vom Typ *UNNotificationSettings*

Listing 12: Deklaration weiterer Funktionen

```
func getNotificationSettings(completionHandler:
@escaping (UNNotificationSettings) -> Void)
func getNotificationCategories(completionHandler:
@escaping (Set<UNNotificationCategory>) -> Void)
func getPendingNotificationRequests
(completionHandler: @escaping
([UNNotificationRequest]) -> Void)
func removePendingNotificationRequests
(withIdentifiers identifiers: [String])
func removeAllPendingNotificationRequests()
func getDeliveredNotifications(completionHandler:
@escaping ([UNNotification]) -> Void)
func removeDeliveredNotifications(withIdentifiers
identifiers: [String])
func removeAllDeliveredNotifications()
```

Link zum Thema

- Referenz des UserNotifications-Frameworks
  https://developer.apple.com/reference/usernotifications

als Teil des Closure-Parameters *completionHandler*. Über dieses Objekt können Sie ermitteln, welche Einstellungen Ihrer App in Bezug auf Notifications zur Verfügung stehen. Dazu bringt dieser Typ Properties für die verschiedenen zur Verfügung stehenden Einstellungen und Status mit, die Sie auslesen und überprüfen können.

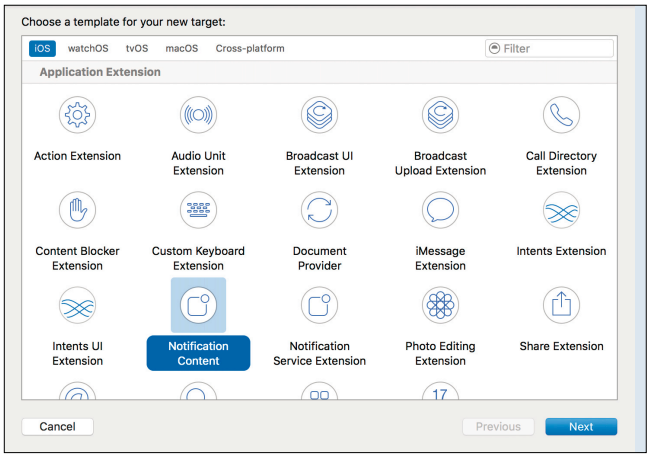
Die nächste Methode *getNotificationCategories(completionHandler:)* gibt Ihnen Zugriff auf die Notification-Categories, die innerhalb Ihrer App gesetzt sind. Dazu liefert diese Methode in ihrem *completionHandler*-Parameter ein Set von UNNotificationCategory-Objekten; je ein UNNotificationCategory-Objekt für jede registrierte Notification-Category. So können Sie jederzeit die gesetzten Kategorien in Ihrer App auslesen.

getPendingNotificationRequests (completionHandler:) liefert Ihnen im Parameter *completionHandler* ein Array an UNNotificationRequest-Objekten. Es handelt sich dabei um die Notification-Requests, die bereits im Singleton des UNUserNotificationCenters registriert sind, aber noch nicht ausgeliefert wurden und somit noch auf Zustellung warten. In diesem Zusammenspiel steht auch die Methode *removePendingNotificationRequests(withIdentifiers:)* zur Verfügung, über die Sie bereits registrierte Notification-Requests noch vor der Zustellung wieder abbrechen und somit eine Zustellung an den Nutzer verhindern können.

Dabei erwartet die Methode ein Array von Strings, wobei ein String dem eindeutigen Identifier des zugehörigen Notification-Requests entspricht, der abgebrochen werden soll. Alternativ steht auch noch die Methode *removeAllPendingNotificationRequests()* zur Verfügung, über die Sie direkt alle registrierten, aber noch nicht zugestellten Notification-Requests abbrechen können.

Zu guter Letzt ist da noch die Methode *getDeliveredNotifications (completionHandler:)*. Sie übergibt Ihnen in deren *completionHandler*-Parameter ein Array von UNNotification-Objekten. Es handelt sich dabei um jene Notifications, die bereits zugestellt wurden, aber noch im Notification Center auf dem Gerät des Nutzers existieren und dort angezeigt werden. Sie können dann die Methode *removeDeliveredNotifications(withIdentifiers:)* dazu nutzen, solche im Notification Center vorhandenen Notifications zu entfernen, indem Sie ihr als Parameter ein Array von Strings übergeben, wobei ein String dem Identifier des zugehörigen Notification-Requests der Notification entspricht, die entfernt werden soll.

Notifications haben ab iOS 10 eine neue optische Aufmachung und können deutlich mehr Informationen enthalten und darstellen (Bild 3)



Mit den neuen Extensions Notification Content und Notification Service Extension lassen sich die Inhalte von Notifications weiter optimieren und anpassen (Bild 4)

Fazit

In diesem Beitrag haben Sie einen Überblick über das neue UserNotifications-Framework erhalten. Mit diesem Framework besitzen Notifications endlich eine plattformübergreifende, einheitliche Basis, die vom UIKit-Framework entkoppelt ist und darüber hinaus über viele mächtige und neue Funktionen verfügt. Gleichzeitig fällt der Umstieg vom alten System nicht allzu schwer, da die grundlegende Funktionsweise und die zugehörigen Befehle auch im UserNotifications-Framework noch immer sehr ähnlich sind.

Weitere spannende Funktionen des Frameworks sind die ausgebauten Multimedia-Möglichkeiten (Bild 3). In diesem Zuge stehen in iOS auch zwei exklusive neue Extensions zur Verfügung, um das damit zusammenhängende Verhalten zu optimieren beziehungsweise zu beeinflussen (Bild 4).

Ich persönlich kann jedem iOS-, tvOS und watchOS-Entwickler nur empfehlen, sich schnellstmöglich mit den Funktionen und den APIs des UserNotifications-Frameworks vertraut zu machen und es in eigenen Apps zu adaptieren. ■

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

IOS-FRAMEWORKS

Ausgelagert

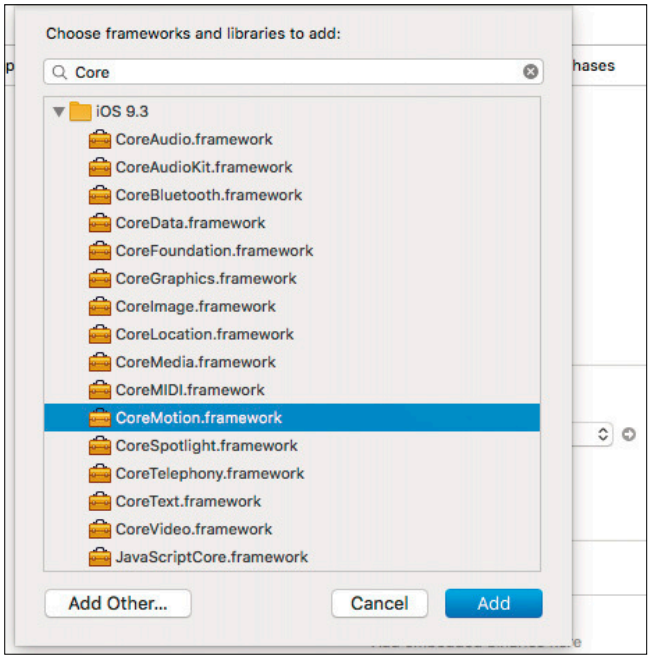
Die Wiederverwendbarkeit von Code ist eines der Hauptmerkmale der OOP.

Die Wiederverwendbarkeit von Code ist in der Regel heutzutage Standard. Allerdings unterscheiden sich die Möglichkeiten bei der Umsetzung etwas von Sprache zu Sprache. Java kennt für die Wiederverwendung Packages, im .NET-Framework wird Code, der noch einmal verwendet werden soll, in entsprechenden Assemblies untergebracht.

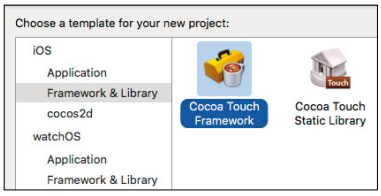
Natürlich gibt es auch für Swift ein entsprechendes Konzept. Wenn Sie bereits die eine oder andere App für die Apple-Plattformen entwickelt haben, dann werden Sie wahrscheinlich auch schon mit diesem Konzept in Berührung gekommen sein. Die Rede ist von den als Framework bezeichneten Bibliotheken unter iOS beziehungsweise macOS, die Sie vielleicht schon einmal in einem oder mehreren Ihrer Projekte eingebunden haben.

In Bild 1 sehen Sie den Arbeitsschritt zur Einbindung eines der iOS-Frameworks, die Bestandteil der ausgelieferten iOS-Version sind. In ein Projekt einbinden können Sie ein solches Framework (zum Beispiel das CoreMotion-Framework), indem Sie im Project Navigator von Xcode das Projekt markieren und dann im Register *General* von Xcode den Abschnitt *Linked Frameworks and Libraries* suchen. Über den +-Button wird das Auswahlfenster *Choose frameworks and libraries to add* geöffnet.

Durch Eingabe von Bestandteilen des Namens im Suchfeld kann die vorhandene Auswahl eingeschränkt werden. Wird das gesuchte Framework aufgelistet, so kann es nach Markierung und einem Mausklick auf den *Add*-Button dem Projekt hinzugefügt werden. Zur Verwendung der im Framework enthaltenen Klassen muss das Framework noch in der Klasse, in welcher der Code verwendet werden soll, via *Import*-Anweisung referenziert werden:



Einbindung eines Frameworks in ein iOS-Projekt (Bild 1)



Projektvorlage für ein iOS-Framework (Bild 2)

```
import UIKit
import CoreMotion

class ViewController: UIViewController {
  override func viewDidLoad() {
    super.viewDidLoad()
  }
}
```

Anschließend können die im Framework vorhandenen Klassen verwendet werden, indem ein Objekt dieser Klassen erzeugt wird.

Ein eigenes Framework anlegen

Um ein Framework selbst anlegen zu können, gibt es in Xcode eine passende Projektvorlage (Bild 2). Zuerst muss die gewünschte Plattform ausgewählt werden. In Xcode wird man unter *iOS, Framework & Library* dann fündig. Hier befinden sich nur zwei Vorlagen zur Auswahl: *Cocoa Touch Framework* und *Cocoa Touch Static Library*.

Nach Markierung des Typs (im Beispiel: *Cocoa Touch Framework)* und Eingabe eines Namens wird das Projektgerüst automatisch erzeugt. Betrachtet man das Projekt im Project Navigator von Xcode, so fällt als Erstes die eingefügte Header-Datei *(*.h)* auf (Listing 1). Dateien dieses Typs kennt man eigentlich nur aus Projekten, die Objective-C als Sprache verwenden.

Mittels der Header-Datei werden die enthaltenen Funktionen des Frameworks nach außen publiziert. Außerdem kann hier eine Versionsnummer für das Framework hinterlegt werden, die sich später auch abfragen lässt.

Code im Framework

Im nächsten Schritt muss die Klasse, die mittels des Frameworks zur Verfügung gestellt werden soll, implementiert wer-

Listing 1: Header-Datei des Frameworks

```
//
//  BspFramework.h
//  BspFramework
//
//  Created by Christian Bleske on 28.06.16.
//  Copyright © 2016 Christian Bleske.
//  All rights reserved.
//

#import <UIKit/UIKit.h>

//! Project version number for BspFramework.
FOUNDATION_EXPORT double BspFrameworkVersionNumber;

//! Project version string for BspFramework.
FOUNDATION_EXPORT const unsigned char
BspFrameworkVersionString[];

// In this header, you should import all the public
// headers of your framework using statements like
// #import <BspFramework/PublicHeader.h>
```

Listing 2: Die Klasse Person

```
//  Person.swift
//  BspFramework

import Foundation

public class Person {

  var Nachname:String
  var Vorname:String

  public init (nachname:String, vorname:String){
    print("Klasse wurde initialisiert!")
    self.Nachname = nachname
    self.Vorname = vorname
  }

  public func ausgabeName(){
    print("Mein Name lautet: \(Nachname),
    \(Vorname)")
  }
}
```

den. Hierzu wird das Projekt erst einmal um eine zusätzliche Swift-Datei ergänzt. Über das *New File*-Menü in Xcode wählen Sie hierzu im Abschnitt *iOS, Source* die Vorlage *Swift File* aus. Nach Eingabe eines Namens wird die neue Datei im Projekt erzeugt. Anschließend kann innerhalb der Datei eine Klasse angelegt werden. Im Beispiel wird dort die Klasse *Person* definiert (Listing 2).

Test des Frameworks

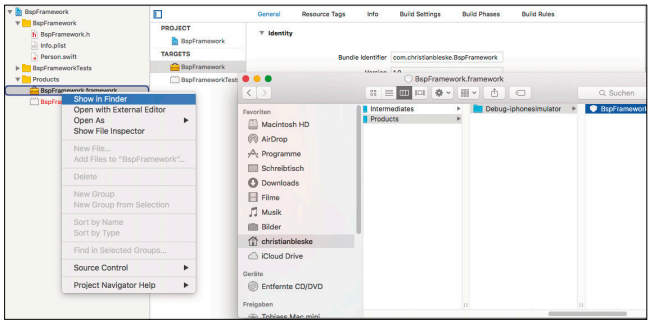
Die Klasse *Person* nimmt im Initialisierer der Klasse zwei Parameter (Nachname, Vorname) entgegen. Über die Methode *ausgabeName* kann der Inhalt der zuvor initialisierten Variablen wieder ausgegeben werden.

Wichtig ist, dass sowohl die Klasse selbst als auch deren Methoden mit dem Schlüsselwort *public* gekennzeichnet werden. Wird die Klasse nicht mit *public* gekennzeichnet, so ist sie außerhalb des Frameworks nicht sichbar und somit natürlich auch nicht ansprechbar.

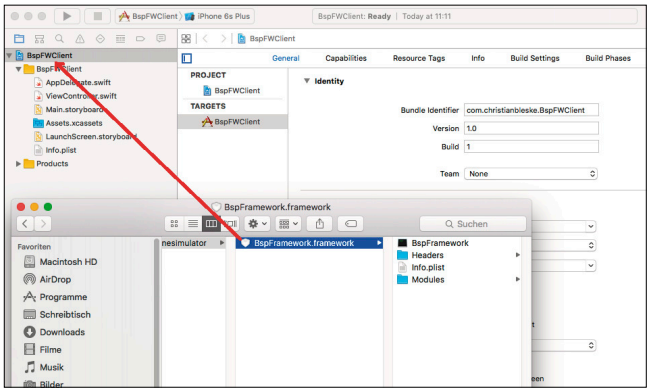
Die vorgestellte Basisfunktionalität genügt für einen ersten Test des Frameworks. Hierzu muss aber zuerst geklärt werden, wie das Framework in ein anderes Projekt eingebunden werden kann. Das Hinzufügen einer Referenz, wie zu Beginn vorgestellt, funktioniert für ein selbst erstelltes Framework leider nicht, da das Framework natürlich nicht Bestandteil der mit Xcode ausgelieferten iOS-Bibliotheken ist.

Nach der Kompilierung stellt sich also zuerst die Frage, wo sich das Framework befindet. Um den Standort der Bibliothek zu erfahren, muss man im Project *Navigator* von Xcode den Ordner *Products* erweitern.

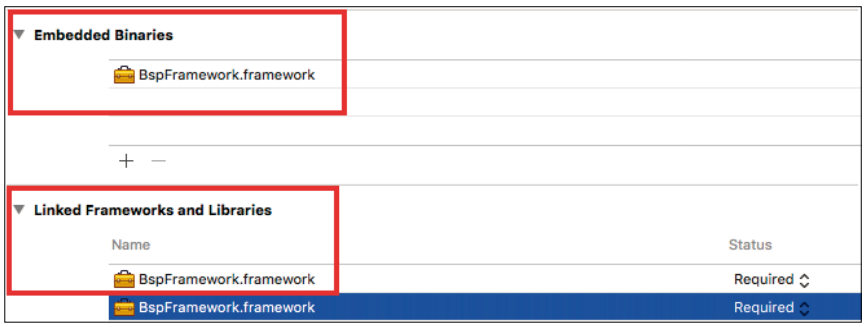
Dort befinden sich zwei Einträge: *Name.framework* und *NameTest.xctest*. Markiert man im Beispiel den Eintrag *BspFramework.framework* und ruft anschließend im Kontextmenü den Menüpunkt *Show in Finder* auf, so wird die nach Kompilierung erstellte Framework-Datei angezeigt (Bild 3). ▶



Erzeugtes Framework im Finder anzeigen (Bild 3)



Framework via Drag and Drop im Projekt einfügen (Bild 4)



Konfiguration des Frameworks im Zielprojekt (Bild 5)

Um das Framework in einem anderen Projekt nutzen zu können, muss dessen Datei dort eingebunden werden. Im nächsten Schritt muss also ein neues Projekt (im Beispiel ist es eine Single View Application mit dem Namen *BspFWClient)* erstellt werden. Nach dem Anlegen wird zuerst das Framework eingebunden. Darzu wird die im Finder angezeigte Framework-Datei via Drag and Drop in das Projekt gezogen (Bild 4). Das ist aber nur die halbe Miete, eine weitere Einstellung fehlt noch. Als Nächstes muss die Projektdatei markiert und dann anschließend die Target-List selektiert werden. Im Register *General* sucht man nun den Abschnitt *Embedded Binaries*. Über den +-Button muss die Framework-Datei ebenfalls dem Projekt hinzugefügt werden. Gegebenenfalls wurde dabei im Abschnitt *Linked Frameworks and Libraries* das Framework zweimal eingetragen – einmal genügt (Bild 5). Löschen Sie ganz einfach einen der Einträge.

Framework und Client

Wenn die Konfiguration erfolgt ist, dann kann die Implementierung des Clients erfolgen. In der App soll eine Instanz der Klasse *Person* erzeugt werden, diese ist im Framework verfügbar. Wichtig ist, dass auch im Code nicht vergessen wird, dass das Framework via *Import*-Anweisung eingebunden werden muss (Listing 3).

Die Implementierung ist sehr einfach gehalten. Innerhalb der Methode *viewDidLoad* wird eine Instanz der Klasse *Person* erzeugt und es werden die beiden Parameter *vorname* und *nachname* übergeben. Anschließend wird in der folgenden Zeile die Methode *ausgabeName* der Klasse *Person* aufgerufen. Wird die Anwendung danach gestartet, so wird im Output-Window der komplette Satz ausgegeben (Bild 6).

Die Client-App hat im Beispiel zur Ausgabe des Namens auf den entsprechenden Code in der selbst geschriebenen Framework-Datei zurückgegriffen. Gelegentlich kann es sein, dass auf Grund eines Bugs in Xcode die Framework-Datei im Projekt nicht gefunden wird. In einem solchen Fall muss der Suchpfad zur Framework-Datei im Projekt überprüft beziehungsweise neu eingebunden werden. Hierzu selektiert man innerhalb der Targets-List das Projekt und öffnet das Register *Build Settings*. Im Suchfeld gibt man dann den Begriff *Framework Search Path* ein. Anschließend werden die Einstellungen zum Suchpfad angezeigt. Ein Doppelklick auf den Suchpfad öffnet ein zusätzliches Fenster. In diesem kann der Suchpfad nun überprüft und gegebenenfalls korrigiert werden.

```
Klasse wurde initialisiert!
Mein Name lautet: Bleske, Christian
()

All Output
```

Ausgabe im Output-Window (Bild 6)

Listing 3: Implementierung des Clients

```
import UIKit
import BspFramework

class ViewController: UIViewController {

  override func viewDidLoad() {
    super.viewDidLoad()

    let myPerson = Person(nachname:
      "Bleske",vorname:"Christian")
    print(myPerson.ausgabeName())

  }

  override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
  }
}
```

Fazit

Die Wiederverwendbarkeit von Code kann unter iOS mit Hilfe eines Frameworks realisiert werden. Ein Framework unter iOS ist eine separate Datei, die ausgewählte Funktionen enthält, welche in weiteren Projekten verwendet werden sollen beziehungsweise können.

Mit ein wenig Aufwand lässt sich ein solches Framework unter iOS auch selbst erstellen, sodass der selbst geschriebene Code projektübergreifend verwendet werden kann. ■

Christian Bleske
ist Autor, Trainer und Entwickler mit dem Schwerpunkt Client/Server und mobile Technologien. Sein Arbeitsschwerpunkt liegt auf Microsoft-Technologien.
cb.2000@hotmail.de

Jetzt kostenlos testen!

Das Fachmagazin für IT-Entscheider

2 Ausgaben kostenlos testen. Mit exklusivem Zugang zu unseren Digitalausgaben. Business-Newsletter inklusive.

www.com-magazin.de/gratis

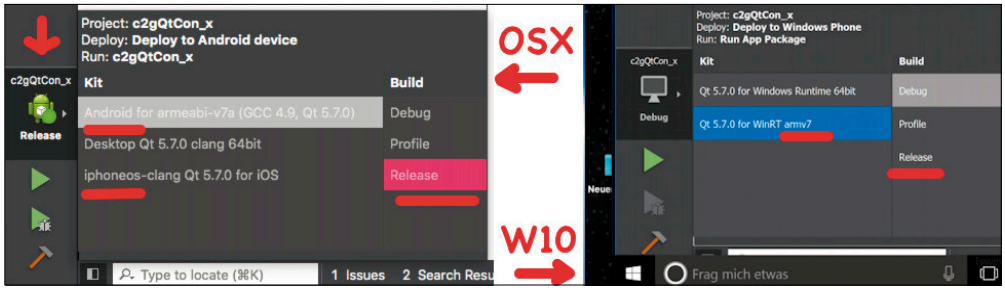
MOBILE APPS CROSS-PLATFORM ENTWICKELN MIT QT 5.7

App in den Store

So bringt man Qt-Quick-Controls-2-Apps in die Stores bei Apple, Google oder Windows.

In den vorangegangenen drei Ausgaben der **web & mobile developer** habe ich ausführlich über die neuen leichtgewichtigen Qt Quick Controls 2 berichtet.

Das Debugging am Gerät oder Simulator hat einfach funktioniert – es fehlte aber der letzte Schritt: das Bereitstellen der Apps in den jeweiligen Stores.



Release Build auswählen (Bild 1)

Release Build versus Debug Build

Bisher haben wir alles immer direkt als Debug Release auf das jeweilige Device deployt. Um nun einen Release zu bauen, müssen wir im Qt Creator den Buildprozess umschalten. Das erledigen wir durch eine Auswahl unten links. In den letzten Folgen habe ich nur über Android- und iOS-App-Entwicklung mit Qt 5.7 gesprochen – aber es können auch Windows-10-Apps für Windows 10 Mobile gebaut werden.

Was unter Windows speziell zu beachten ist, werde ich in einem extra Beitrag genauer ausleuchten – hier beziehe ich mich jetzt nur auf den Buildprozess, um Releases zu bauen.

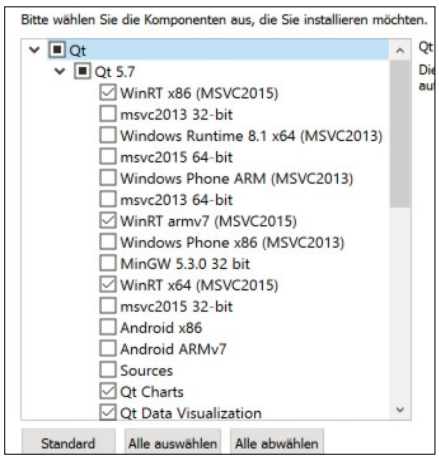
Im Bild 1 ist zu sehen, dass ich unter OS X für Android und iOS Apps entwickle. Um Windows-10-Apps zu bauen, habe ich mir in einer Parallels-VM ein Windows 10 und Visual Studio 2015 (Community Edition) installiert. Wichtig ist, dass man *universal app development* bei der Visual-Studio-Installation auswählt, damit die richtigen Compiler zur Verfügung stehen.

Danach habe ich in diesem Windows eine normale Qt-5.7-Installation vorgenommen und dort drei Kits für Visual Studio 2015 ausgewählt, nämlich *winrt-x86, winrt-x64* und *winrt-arm-msvc2015* (Bild 2). Damit sind Qt Creator und Visual Studio vorbereitet.

Meine Projekte sind alle bei GitHub – daher klone ich das Projekt dann ebenfalls in der Windows-Installation, um es dort für Qt Creator nutzen zu können. Da ich unter OS X auch auf die Windows-Dateiverzeichnisse zugreifen kann, nehme ich zum Klonen mein OS-X-GitHub-Tool (Tower für Github)

Nun zurück zum Qt Creator. Dort wählen wir die Zielplattform aus: *Android, iPhone, WinRT arm v7* sowie *Release*. Um den Buildprozess zu konfigurieren, müssen wir links bei den Auswahlen die Projektansicht anklicken (Bild 3).

Qt-Creator-Projekt: Ansicht wählen (Bild 3)



Qt-Creator-Pakete für Windows 10 (Bild 2)

Android Release Build Settings (Bild 4)

Außerdem werden je nach Plattform Änderungen in der Projektbeschreibungsdatei *(*.pro)* notwendig, und es müssen Dateien in jeweils OS-spezifische Ordner kopiert werden. Das betrachten wir jetzt für jede Plattform getrennt. Fangen wir dazu mit Android an – dort ist der gesamte Prozess am einfachsten.

Android Release Build

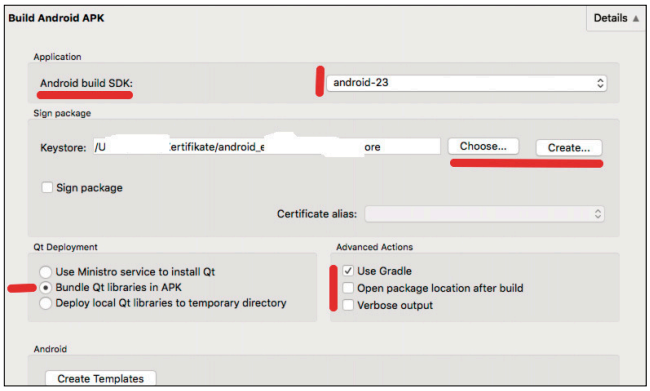
Über die Projektansicht können wir den Android-Release konfigurieren. Bild 4 zeigt die möglichen Einstellungen für den Release Build. Da zum Schluss eine Menge Dateien beim Release erzeugt werden, sollte *Shadow Build* angekreuzt sein. Das bewirkt, dass alle Dateien in das angegebene Build-Verzeichnis geschrieben werden. Das Build-Verzeichnis wird gemäß den allgemeinen Einstellungen vorgeschlagen und kann geändert werden. Der Shadow Build sorgt auch dafür, dass keine Builddaten aus Versehen ins Versionskontrollsystem gelangen.

Anschließend klicken Sie dann im Fenster unter *Build Android APK* auf *Details*. Bild 5 zeigt die APK-Konfiguration. Bestimmen Sie das Android Build SDK – dies sollte eines der neuesten SDKs sein – in meinem Fall *android-23* für Android Marshmallow.

Um Releases bauen zu können, benötigen Sie ein Zertifikat. Wenn Sie schon einen Android-Keystore haben, wählen Sie bitte den entsprechenden aus. Sie können aber über den Button *Create …* auch direkt aus dem Qt Creator heraus einen neuen Keystore erzeugen.

Die Signierung selbst nehmen wir erst vor, wenn wir den Release bauen – dazu haben wir aber noch nicht alle Artefakte zusammen.

Unter *Android* finden Sie einen Button *Create Templates*. Darüber werden innerhalb des Projekts einige Dateien erzeugt – unter anderem als Wichtigstes eine von Qt vorkonfigurierte Manifestdatei. Bild 6 zeigt die erzeugte Dateistruktur und die Dateien. Alle eigenen App-Icons sollten Sie jetzt in die entsprechenden Ordner kopieren. Diese Struktur kann man beliebig erweitern, um beispielsweise Icons für weitere



Qt Android APK Settings (Bild 5)

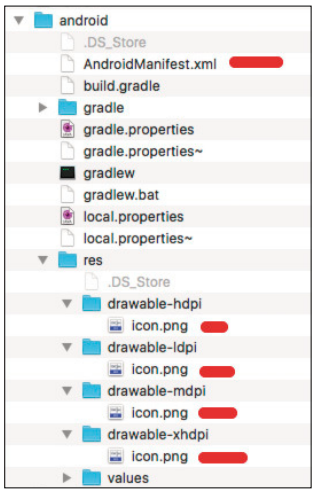
Auflösungen hinzuzufügen. In der *.pro*-Datei wurde Folgendes automatisch ergänzt:

```
DISTFILES += \
...
android/AndroidManifest.xml \
android/gradle/wrapper/gradle-wrapper.jar \
android/gradlew \
android/res/values/libs.xml \
android/build.gradle \
android/gradle/wrapper/gradle-wrapper.properties \
android/gradlew.bat

ANDROID_PACKAGE_SOURCE_DIR = $$PWD/android
```

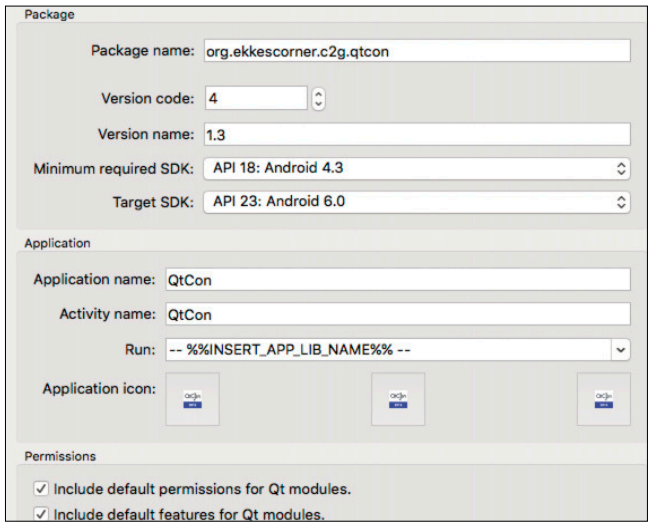
Die Angabe des *ANDROID_PACKAGE_SOURCE_DIR* verweist auf das Projektverzeichnis */android* und kopiert alles, was dort gefunden wird, mit in das APK.

Das erzeugte Manifest kann man direkt aus Qt Creator mit einem speziellen Editor bearbeiten – das ist also sehr komfortabel gelöst (Bild 7). Hier tragen Sie jetzt Ihren eindeutigen Package-Namen ein. Alle Beispiele in diesem Artikel beziehen sich auf eine App von mir, die als Open Source bei GitHub verfügbar ist: die Qt-Con 2016-App für die gleichnamige Qt-Community-Konferenz (http://github.com/ekke/c2gQtCon_x).

In der Manifestdatei passen Sie das Minimum und das Target SDK an. Qt 5.7 unterstützt alle Android-Versionen ab 4.3 (API 18). Für die drei gängigsten Auflösungen können Sie auch schon Programm-Icons zuordnen. Bei den Zugriffsberechtigungen versucht Qt, anhand der genutzten Libraries die richtigen Permissions in das Manifest einzutragen – das können Sie ▶



Das Android-Verzeichnis des Qt-Projekts (Bild 6)

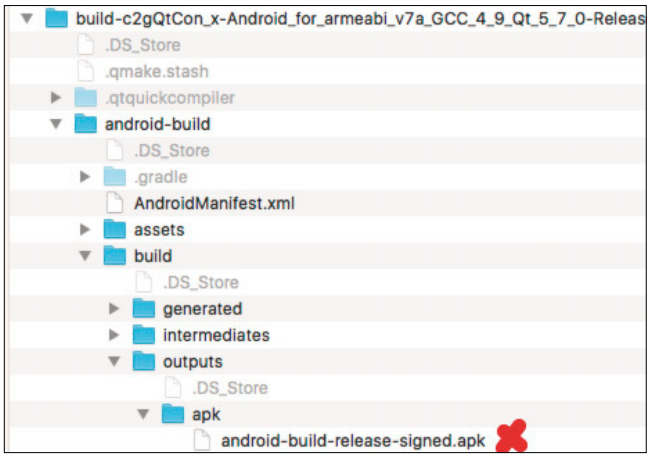


Der Android Manifest Editor von Qt Creator (Bild 7)

aber fein granular anpassen. Jetzt haben wir alle Voraussetzungen, um den Release Build zu starten. Gehen Sie wieder zurück zu den Build Details für APK und klicken Sie dort auf *Sign Package*. Dann müssen Sie das Passwort angeben. Als Nächstes klicken Sie unten links auf den Build-Button. Jetzt baut Qt das APK und signiert es.

Bild 8 zeigt den Ordner, in dem sich dieses APK befindet. Wichtig: Die Datei muss *android-build-release-signed.apk* heißen. Nur dann sind Sie sicher, dass es sich um einen Release Build handelt und dieser auch signiert wurde. Jetzt können Sie dieses APK zu den Android-App-Stores Ihrer Wahl hochladen. Meine QtCon-2016-App finden Sie bei Google Play und im Amazon App Store.

Der Android-Buildprozess findet also komplett innerhalb von Qt Creator statt und Sie benötigen keine Android-IDE.



Das Ausgabeverzeichnis für den Android Release Build (Bild 8)

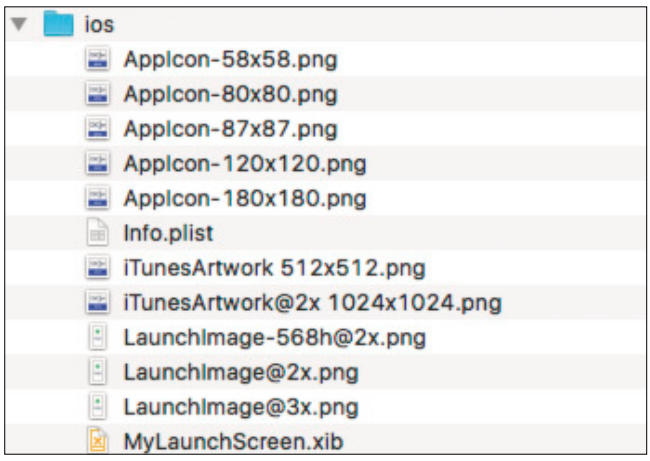
iOS Release Build

Bei iOS ist das anders. Xcode muss auf dem Rechner installiert sein. Sie benötigen also zumindest einen Mac-OS-Rechner. Außerdem müssen Sie sich beim Apple Developer Program registriert haben und ein entsprechendes Zertifikat bekommen haben. Testen Sie, ob Sie ein Beispielprojekt in Xcode erzeugen können, bevor Sie in Qt Creator nach Fehlern suchen.

Qt Creator erzeugt ein Xcode-Projekt, aus dem Sie dann die App für den App-Store generieren. In den bisher durchgeführten Debug-Builds wurde auch schon ein Xcode-Projekt erzeugt. Gehen Sie in das entsprechende Verzeichnis und suchen die *Info.plist*-Datei. Diese Datei kopieren Sie in das Qt-Creator-Projektverzeichnis.

Nun müssen Sie in der *Info.plist*-Datei projektspezifische Anpassungen vornehmen. Dazu ist leider etwas Fachwissen nötig, das heißt, Sie sollten die Apple-Developer-Dokumentation zurate ziehen, um zu erfahren, was Sie bei den entsprechenden Keys eintragen müssen. Hier einige der Keys, um den eindeutigen Package-Namen, den App-Namen und die Version sowie die Device Family einzutragen:

```
<key>CFBundleIdentifier</key>
<string>org.ekkescorner.c2g.qtcon</string>
```



Das iOS-Verzeichnis des Qt-Projekts (Bild 9)

```
<key>CFBundleDisplayName</key>
<string>QtCon</string>
<key>CFBundleName</key>
<string>QtCon</string>
<key>CFBundleShortVersionString</key>
<string>1.3</string>
<key>CFBundleVersion</key>
<string>1.3</string>
<key>LSRequiresIPhoneOS</key>
<true/>
```

Dann sind in der *Info.plist*-Datei die ganzen Angaben für App-Icons, Store-Icons und Launch-Screens zu machen. Schauen Sie sich im GitHub-Projekt an, was ich dort zugewiesen habe. Wichtig zu wissen ist, dass ab dem iPhone 6 der Launch-Screen keine Imagedatei mehr ist, sondern eine Storyboard-Datei. Diese generiert Ihnen Qt bereits – trägt dort aber den Target-Namen ein.

Das Einfachste ist nun, die Storyboard-Datei ins Verzeichnis */ios* zu kopieren und dort im Quelltext den Text auszutauschen. Aber schön sieht das immer noch nicht aus. Ich habe daher in Xcode selbst diese *.xib*-Datei editiert und dort ein Image eingebaut, das im Launch-Screen angezeigt wird. Sie finden diese Datei im GitHub-Projekt unter */ios/MyLaunchScreen.xib*. Der zugehörige Eintrag in der *Info.plist* lautet:

```
<key>UILaunchStoryboardName</key>
<string>MyLaunchScreen</string>
```

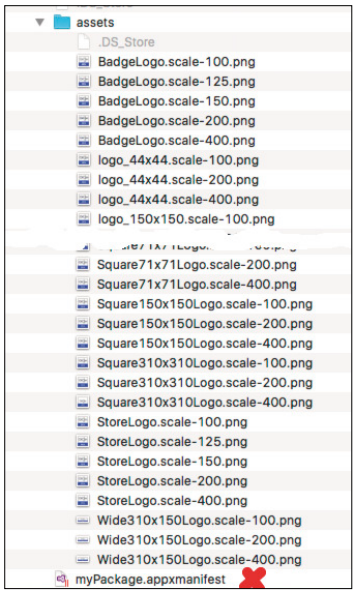
In der *.pro*-Datei werden ebenfalls ein paar Informationen für den Qt-Buildprozess benötigt:

```
ios {
  QMAKE_INFO_PLIST = ios/Info.plist
  ios_icon.files = $$files($$PWD/ios/AppIcon*.png)
  QMAKE_BUNDLE_DATA += ios_icon
  ios_artwork.files =
  $$files($$PWD/ios/iTunesArtwork*.png)
  QMAKE_BUNDLE_DATA += ios_artwork
  app_launch_images.files =
```

```
  $$files($$PWD/ios/LaunchImage*.png)
  QMAKE_BUNDLE_DATA += app_launch_images
  app_launch_screen.files =
  $$files($$PWD/ios/MyLaunchScreen.xib)
  QMAKE_BUNDLE_DATA += app_launch_screen

  QMAKE_IOS_DEPLOYMENT_TARGET = 8.2
  # Note for devices: 1=iPhone, 2=iPad, 1,2=Universal.
  QMAKE_IOS_TARGETED_DEVICE_FAMILY = 1
}
```

Jetzt werden die entsprechenden Dateien mit in das erzeugte Xcode-Projekt kopiert. Innerhalb des Qt-Creator-Projekts sieht das */ios*-Verzeichnis in etwa so aus wie in Bild 9.

Wie gewohnt wählen Sie unten links *Release* und *iOS* aus, achten dann darauf, dass ein iPhone per USB angeschlossen ist, und klicken auf *Build*. Dann suchen und öffnen Sie das entsprechende generierte Xcode-Projekt. Dort ist das Projekt bereits komplett konfiguriert (Bild 10). Der Aufruf von *Projekt, Archive* baut den Release, den Sie dann direkt zu iTunes Connect hochladen können. Achtung: Package-Name und Device-Family müssen mit Ihrem iTunes-Connect-Projekt übereinstimmen.



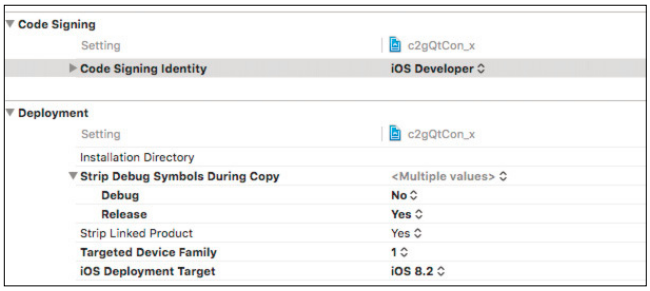
Das WinRT-Verzeichnis des Qt-Projekts (Bild 11)

Wenn man sich einmal ein wenig in die Keys der *Info.plist* eingearbeitet hat, ist der gesamte Workflow auch für iOS-Apps unter Qt 5.7 aus Qt Creator heraus einfach und intuitiv.

Das kann man vom derzeitigen Stand des Windows-10-Buildprozesses noch nicht behaupten. Aber das ist ja auch erst ganz am Anfang. Für Windows 10 benötigt die *.pro*-Datei einige Angaben:

```
winrt {
  WINRT_MANIFEST.name = Qt Con 2016
  WINRT_MANIFEST.background = white
  WINRT_MANIFEST.default_language = en
  WINRT_MANIFEST.description = QtCon 2016 Conference App
  WINRT_MANIFEST.version = 1.3.0.0
  WINRT_MANIFEST = winrt/myPackage.appxmanifest
}
```

Geben Sie hier den Programmnamen, eine Beschreibung und die Version ein. Qt erzeugt im Buildprozess ein Visual-Studio-2015-Projekt. In Visual Studio können Sie dann die Manifestdatei anpassen und diese Datei in das Qt-Projekt unter */winrt* kopieren. Es gibt noch keinen Button, der das automatisch erledigt. Die ganzen benötigten Programm-Icons fügen Sie in Visual Studio hinzu. Visual Studio packt diese in den Ordner */assets* und benennt die Dateien nach einem eigenen Naming-Schema um. Danach kopieren Sie nicht nur die Manifestdatei, sondern auch die Images aus Visual Studio in das Qt-Projekt. Bild 11 zeigt die Qt-Creator-Projektstruktur.



Das Xcode-Projekt, aus Qt Creator erzeugt (Bild 10)

Der Build selbst muss über die *cmd.exe* angestoßen werden. Hier sind die entsprechenden Commands:

```
C:\Program Files (x86)\Microsoft Visual Studio 14.0\vc\
vcvarsall
cd C:\xxxxx\build-c2gQtCon_x-w10
qmake -tp vc ..\c2gQtCon_x\c2gQtCon_x.pro
```

vcvarsall setzt Pfade, die Visual Studio benötigt. Gehen Sie dann ins Buildverzeichnis und rufen Sie *qmake* auf. Das erzeugt das Visual-Studio-Projekt, aus dem Sie die App in den Windows App Store laden können. Gegebenenfalls müssen Sie nach der Generierung noch die Dateien aus dem Qt-Projektordner */winrt/assets* in den Visual-Studio-Ordner */assets* kopieren.

Fazit

Qt 5.7 Quick Controls 2 ermöglichen es nicht nur, gut aussehende Programme zu bauen. Qt gibt auch Unterstützung im Release Build, um die Apps für den Upload in die diversen Stores vorzubereiten. In meinem Open-Source-Programm habe ich dann zunächst einen Betatest gestartet. Diese plattformspezifischen Betatest-Programme machen es einfach, Tests zu organisieren.

Das Deployment bei Google ist sehr schnell. Bei Apple ist es normalerweise innerhalb von 24 Stunden so weit. ■

Ekkehard Gentz
ist Autor, Trainer und Speaker auf Konferenzen. Er entwickelt als Independent Software Architect mobile Anwendungen für internationale Kunden, ist BlackBerry Elite Member und bloggt unter: **http://ekkes-corner.org**

SMART DATA
Developer Conference
Big Data & Smart Analytics

Für Softwareentwickler und IT-Professionals

06. Dezember 2016, Köln

web & mobile DEVELOPER Leser erhalten 15 % Rabatt mit Code SMART16wmd

Themenauswahl:

- Effizienter durch standardisierte Reports
- Datenqualität erhöhen
- Recommender Algorithmen: R vs. Spark
- Streaming = Zukunft von Big Data

Ihre Experten (u.a.):

Damir Dobric
DAENET Corporation

„German Cloud für Entwickler und IT-Pros“

Constantin Klein
Freudenberg IT SE & Co. KG

„Eine Branche im Daten-Goldrausch“

Dr. Hanna Köpcke
Webdata Solutions GmbH

„Datenqualität: Big Data Matching“

Stefan Papp
The unbelievable Machine Company GmbH

„Streaming mit Apache Flink“

Tobias Trelle
codecentric AG

„Einführung in Graphen-Datenbanken mit Neo4j“

smart-data-developer.de #smartddc SMARTDATADeveloperConference

Präsentiert von: dotnetpro web & mobile DEVELOPER

SMART DATA
Developer Conference

Big Data & Smart Analytics **06. Dezember 2016, Köln**

08.45	**Begrüßung**
09.00 – 09.55	**Keynote: Eine Branche im Goldrausch** *Constantin Klein* Die Datenexplosion ist nicht mehr aufzuhalten. Zwischen Hype und Realität entdecken Sie einen praxisorientierten Ansatz, um individuell Ihren Claim in diesem Goldrausch abzustecken.
10.00 – 10.55	**Datenqualität: Big Data Matching** *Dr. Hanna Köpcke* Im Rahmen dieses Vortrages werden Herausforderungen beim Matching von Big Data beleuchtet und Lösungsansätze auf Basis von Map/Reduce aufgezeigt.
11.30 – 12.25	**4x4: Big Data in der Cloud?** *Danny Linden* Daten in die Cloud auslagern? Warum und wenn ja, bei welchem Provider? Anhand von vier Beispielen können Sie eine geeignete Lösung finden.
12.30 – 13.25	**NoSQL: Einführung in Graphdatenbanken mit Neo4j** *Tobias Trelle* Warum überhaupt nicht-relational? Lernen Sie Neo4j, die populärste Graphen-Datenbank, inkl. Datenstrukturen, Query-Language Cypher und API kennen - eine spezielle NoSQL-DB.
14.00 – 14.30	**Lunch-Session: „German Cloud“ für Devs und IT-Pros** *Damir Dobric und Andreas Erben* Was geht, und was nicht? Was ist anders?
14.30 – 15.25	**Implementierung von Recommender-Algorithmen** *Dr. Henrik Behrens* Anhand eines konkreten Anwendungsfalls, der Programmierung eines Recommender-Systems, vergleichen wir eine konventionelle Implementierung in R mit zwei Implementierungen in der Big-Data-Technologie Spark (Spark DataFrames und Spark MLlib).
16.00 – 16.55	**Smart Analytics: Streaming mit Apache Flink** *Stefan Papp* Lernen Sie Apache Flink kennen und wie man eine Streaming-Applikation damit schreibt. Streamingplattformen sind die Zukunft von Big Data.
17.00 – 18.00	**Visualisierung, Active & Mobile Reports – ein Anwendungsbeispiel** *Holger Gerhards* Welche Vorteile bieten Standardisierungen im Reporting? Sehen Sie, wie Reports in wenigen Minuten mit wenigen Klicks erstellt werden. Entdecken Sie die Vorteile und Möglichkeiten von HICHERT®SUCCESS im IBM Analytics Umfeld.

Programmänderung vorbehalten

Anmeldung: smart-data-developer.de

Veranstalter: developer media — Neue Mediengesellschaft Ulm mbH

FEHLERSUCHE IN ANDROID-APPS

Hirte der Fehler

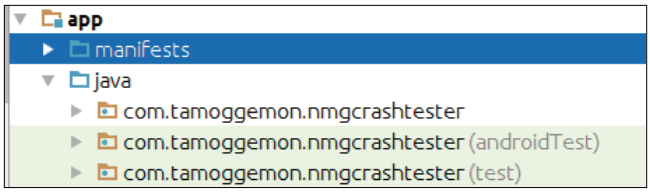
Entwickler von Android-Applikationen können auf eine Vielzahl von automatisierten Systemen zur Fehlermitigierung zurückgreifen.

Bewertungen sind in der heutigen, Rating-getriebenen Handcomputerindustrie alles. Ein fehlerhafter Release reicht aus, um das Rating im Store langfristig zu ruinieren.

Unit-Tests sind vergleichsweise einfach. Es handelt sich dabei um Miniprogramme, die von einem Test Runner ausgeführt werden. Ihre Aufgabe ist das Durchführen von einer oder mehreren charakteristischen Funktionen, deren Ergebnisse sodann mit den vom Entwickler vorgegebenen verglichen werden – tritt eine Differenz auf, so ist etwas faul.

Unit Testing mit Android Studio

Angesichts der Wichtigkeit von Unit Testing stattet Googles aktuelle IDE jedes neu erstellte Android-Projekt automatisch mit zwei Testprojekten aus – Bild 1 zeigt, was Sie nach der Abarbeitung des Projektassistenten erwartet.



Android Studio 2.1 liefert für Entwickler zwei Testprojekte mit (Bild 1)

Unit-Tests werden erfahrungsgemäß immer dann besonders gern ausgeführt, wenn sie nicht sonderlich viel Zeitaufwand erfordern. Diesem Anspruch kann die Nutzung eines vollwertigen Emulators nicht gerecht werden. Google begegnet dem Problem dadurch, dass die Unit-Tests in zwei Gruppen aufgeteilt werden.

Das mit *(test)* bezeichnete Projekt enthält die als *Local JVM Tests* bezeichneten Testfälle. Sie werden von Android Studio unter Nutzung der lokalen JVM des Hosts ausgeführt und sind somit zur Verifikation klassischer Logik geeignet. Im Projekt *(androidTest)* finden sich hingegen all jene Testfälle, die in einem Emulator oder auf echter Hardware abgearbeitet werden müssen.

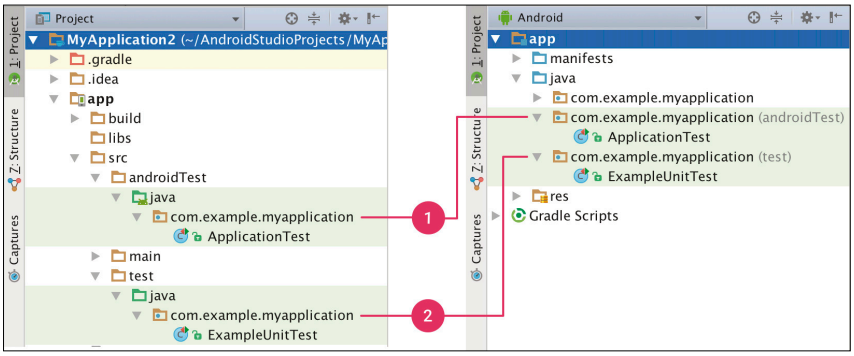
Die beiden Verzeichnisse sind übrigens auch in der normalen Dateiansicht erkennbar. Bild 2 korreliert die Speicherorte.

Bei sorgfältigem Lesen des vorhergehenden Absatzes ist die Rollenverteilung klar: Local JVM Tests lassen sich nur dann ausführen, wenn der zu testende Code vom Rest des Android-Frameworks unabhängig ist. Die JVM des Hosts kann mit Google-spezifischen Klassen naturgemäß nur wenig anfangen.

Ein klassischer Anwendungsfall ist das Prüfen des Verhaltens von Logikklassen. Als Beispiel dafür dient folgende Klasse, die die Summe der Fibonaccizahlen bis zu einem gewissen Rang berechnet – dass der vorliegende Code ob der Rekursion nicht sonderlich effizient ist, ist dem Autor bekannt:

```
public class NMGFib
{
  public static long fib(int n)
  {
    if (n <= 1) return n;
    else return fib(n-1) + fib(n-2);
  }

  public static long collectFibby(int _n)
  {
    long accu=0;
    for(int i=1;i<=_n;i++)
      accu += fib(i);
    return accu;
  }
}
```

Da Abhängigkeiten zu Googles Betriebssystem hier nicht vorliegen, können wir mit einem lokalen Test drauflostesten.



Der Zusammenhang zwischen virtuellen und realen Verzeichnissen ist bei Testfällen oft etwas komplex (Bild 2)

Öffnen Sie die Datei *ExampleUnitTest.java* und passen Sie ihren Inhalt folgendermaßen an:

```
public class ExampleUnitTest {
  @Test
  public void fib_works() throws Exception {
    assertEquals(2, NMGFib.collectFibby(2));
  }
}
```

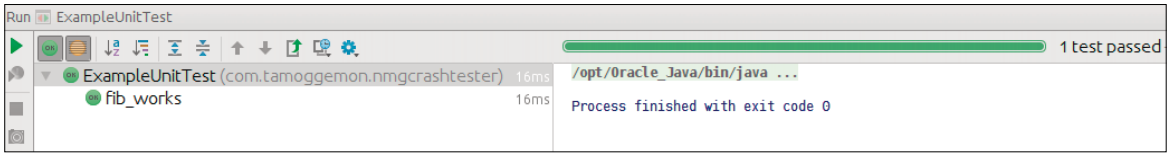
Von besonderem Interesse ist hier die *@Test*-Annotation, die die Routine als für den Testrunner relevant kennzeichnet. Im Rahmen des Starts eines Unit-Test-Laufs erfolgt eine Reflexion, die alle abzuarbeitenden Methoden erreicht.

Wer die hier verwendete *Assert*-Methode reflektiert, bekommt ein Abbild der aus normalem Java bekannten Assertationsklasse zurückgeliefert. Tabelle 1 zeigt einige besonders interessante Funktionen – für eine weitere Besprechung sei auf klassische Java-Lehrbücher verwiesen.

Damit stellt sich die Frage nach der Ausführung des Codes. Android Studio bringt einen Testrunner mit, der für die Abarbeitung von Java-Tests zuständig ist. Er lässt sich durch einen Rechtsklick auf ein Testprojekt oder einen Testfall aktivieren: Wählen Sie im Kontextmenü einfach die Option *Run Tests in <Projektname>*. Android Studio blendet daraufhin die in Bild 3 gezeigte Maske ein, in der die Unit-Tests anhand ihrer Klassenstruktur heruntergebrochen werden.

Tabelle 1: Interessante Funktionen

Funktion	Kurzbeschreibung
static public void fail(String message)	Lässt den Testfall scheitern. Der übergebene String wandert in Richtung des Testrunners.
static public void fail()	Lässt den Testlauf ohne Nachricht scheitern. Die meisten *assert()*-Methoden enthalten ein Overload, das ohne Message auskommt.
static public void assertNotEquals(double unexpected, double actual, double delta)	Vergleicht zwei Fließkommazahlen unter Berücksichtigung eines erlaubten Maximalfehlers.
public static void assertArrayEquals(long[] expecteds, long[] actuals)	Vergleicht zwei Arrays. Diverse Überladungen erlauben das Anpassen an andere Datentypen wie *bool*.
static public void assertNull(Object object)	Nimmt an, dass das Objekt null ist.
static public void assertNotNull(Object object)	Nimmt an, dass das Objekt nicht null ist.



Android Studio assistiert beim Abarbeiten von Testfällen (Bild 3)

Auflösen einfacherer Dependenzen

Abhängigkeiten zum Android-Basisframework stellen kein absolutes Ausschlusskriterium für lokale Unit-Tests dar. Ein in der Literatur als Mocking bezeichneter Prozess stellt einen gangbaren Workaround zum Auflösen einfacherer Dependenzen dar.

Als Beispiel dafür wollen wir eine Methode realisieren, die die zur Abarbeitung notwendigen Informationen aus einem zum Android-Framework gehörenden Objekt bezieht:

```
import android.widget.TextView;
public class NMGFib {
  public static String returner(TextView _aView)
  {
    return _aView.getText().toString();
  }
```

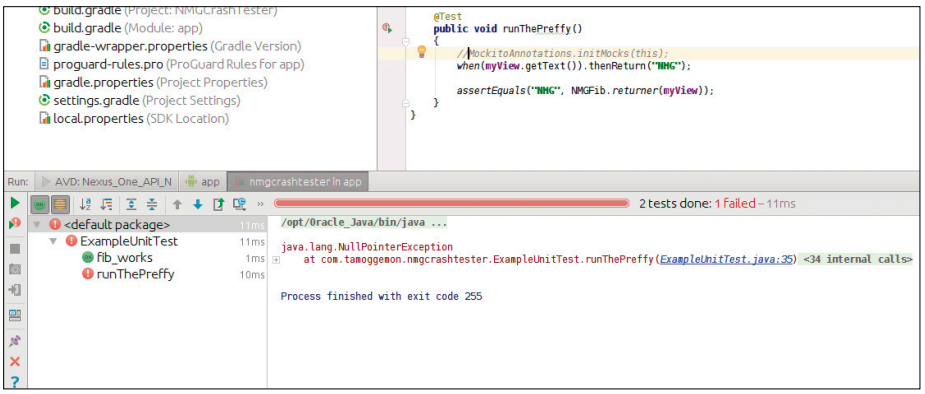
Die hier besprochenen Methoden lassen sich nicht zur Simulation statischer Methoden einsetzen. Weitere Informationen hierzu finden sich unter http://stackoverflow.com/questions/4482315/why-does-mockito-not-mock-static-methods. Ein dazu passender Unit-Test würde folgendermaßen aussehen – wir gehen an dieser Stelle einfach davon aus, dass die TextView immer den Wert *NMG* retourniert:

```
public class ExampleUnitTest {
  @Test
  public void runThePreffy()
  {
    assertEquals("NMG", NMGFib.returner
    (new TextView(null)));
  }
}
```

Java-Kenner nehmen an, dass die Kompilation des Codes aufgrund der Nutzung einer Nicht-Android-VM scheitert. Dies ist allerdings nicht der Fall: Unit-Tests werden mit einer speziellen Variante von *android.jar* ausgeführt, die alle Betriebssystemfunktionen durch Exception werfende Stubs ersetzt. Dies lässt sich durch unseren Unit-Test prüfen: Wer ihn im Test Runner anwirft, erhält von Android Studio die folgende Fehlermeldung, die zudem zum Fehlschlag des betreffenden Testfalls führt:

```
java.lang.RuntimeException: Method getText in android.
widget.TextView not mocked. See http://g.co/android ▶
```

```
studio/not-mocked for details.
at android.widget.TextView.
getText(TextView.java)
at com.tamoggemon.nmgcrashtester.
NMGFib.returner(NMGFib.java:15)
at com.tamoggemon.nmgcrashtester.
ExampleUnitTest.
runThePreffy(ExampleUnitTest.
java:27)
```



Die Testausführung scheitert beim Fehlen von *initMocks* mit einer nichtssagenden Exception (Bild 4)

An dieser Stelle lauert eine kleine Falle: *Android.jar* ist im Bereich der Konstanten relativ vollständig. Die folgende Methode würde auch bei der Ausführung in der lokalen VM keine Exception werfen, weil die Konstanten eins zu eins übernommen wurden:

```
public class NMGFib {
  public static int returner()
  {
    return Color.BLACK;
  }
```

Ein weiteres Ärgernis ist die *toString()*-Methode: Ein Gutteil der in Object enthaltenen Methoden ist auch in der lokalen VM verfügbar, da er ja aus Java stammt und mit Android an sich nichts zu tun hat. Sie lässt sich somit – im Allgemeinen – auch in Testfällen nutzen, die ohne Android-APIs auskommen müssen.

Integration des Mockito-Frameworks

Zur Integration des von Google empfohlenen Mockito-Frameworks ist eine Änderung an der zum Modul *App* gehörenden *build.gradle* notwendig. Passen Sie den Dependencies-Block folgendermaßen an:

```
dependencies {
  compile fileTree(dir: 'libs', include: ['*.jar'])
  testCompile 'junit:junit:4.12'
  compile 'com.android.support:appcompat-v7:24.0.0'
  testCompile 'org.mockito:mockito-core:1.10.19'
}
```

Nach der Änderung ist eine bestehende Internetverbindung erforderlich: Gradle lädt die Mockito-Bibliothek im Rahmen der Resynchronisation automatisch herunter. Im nächsten Schritt muss der Testfall um eine Version von *TextView* ergänzt werden. Diese muss prinzipiell Klassenmember sein – *@Mock* lässt sich nicht auf lokale Variablen anwenden:

```
public class ExampleUnitTest {
  @Mock
  TextView myView;
```

Damit können wir uns der Realisierung von *getText* zuwenden. Mockito bietet mit der unter http://site.mockito.org/mockito/docs/current/org/mockito/Mockito.html im Detail beschriebenen *when()*-Funktion ein Werkzeug an, mit dem die zu emulierenden Methoden samt zu retournierenden Werten bei der Runtime angemeldet werden können:

```
  @Test
  public void runThePreffy()
  {
    MockitoAnnotations.initMocks(this);
    when(myView.getText()).thenReturn("NMG");
    assertEquals("NMG", NMGFib.returner(myView));
  }
}
```

Android ist quelloffen: Wenn die Entwicklung eines Mocks in Arbeit ausartet, spricht nichts dagegen, Googles Arbeiten durch Reflexion zweckzuentfremden.

An dieser Stelle wartet eine kleine Falle auf angehende Entwickler: Mock-Objekte müssen vor der erstmaligen Verwendung durch Aufruf von *MockitoAnnotations.initMocks* bei der Runtime angemeldet werden – unterbleibt dies, so wirft das Framework die in Bild 4 gezeigte Exception.

Testfall-Konstruktor

Da das manuelle Aufrufen von *MockitoAnnotations.initMocks* in jedem Testfall nicht sonderlich befriedigend ist, bietet sich die Nutzung eines Testfall-Konstruktors an. Leider funktioniert *@BeforeClass* in diesem Zusammenhang nicht. Die Nutzung des Makros setzt eine statische Methode voraus, die mit *this* inkompatibel ist:

```
@BeforeClass
public static void initTheMockito()
{
  MockitoAnnotations.initMocks(this); //geht nicht
}
```

Das zur Realisierung von Android-Unit-Tests verwendete Framework bietet einige Annotations an, die die Steuerung des Verhaltens des Test Runners ermöglichen. Eine davon ist *@before*, deren Nutzung im Zusammenspiel mit einer Hilfs-

variablen das mehrfache Aufrufen von *initMocks* verhindert. Ein auf diesem Prinzip aufbauendes Testobjekt präsentiert sich so:

```
public class ExampleUnitTest {
  boolean myInited=false;
  @Mock
  TextView myView;
  @Before
  public void initTheMockito(){
    if(myInited==false){
      myInited=true;
      MockitoAnnotations.initMocks(this);
    }
  }
  @Test
  public void runThePreffy(){
    when(myView.getText()).thenReturn("NMG");
    assertEquals("NMG", NMGFib.returner(myView));
  }
}
```

So nützlich Mocking-Lösungen auch sein mögen: Wer das gesamte Betriebssystem nachbauen muss, stößt an die Grenzen des ökonomisch Sinnvollen. Zudem sorgen statische Methoden für Ärger – in der Praxis kommt man bald an einen Punkt, an dem ein volles Android-Betriebssystem hilfreich ist.

Für die im Folgenden beschriebenen Schritte ist eine zusätzliche SDK-Komponente erforderlich. Starten Sie das SDK-Verwaltungswerkzeug und prüfen Sie, ob das in Bild 5 hervorgehobene Paket auf Ihrer Workstation installiert ist.

Öffnen Sie die zur Applikation gehörende *build.gradle*-Datei und ergänzen Sie sie im ersten Schritt um die Einbindung der folgenden Bibliotheken:

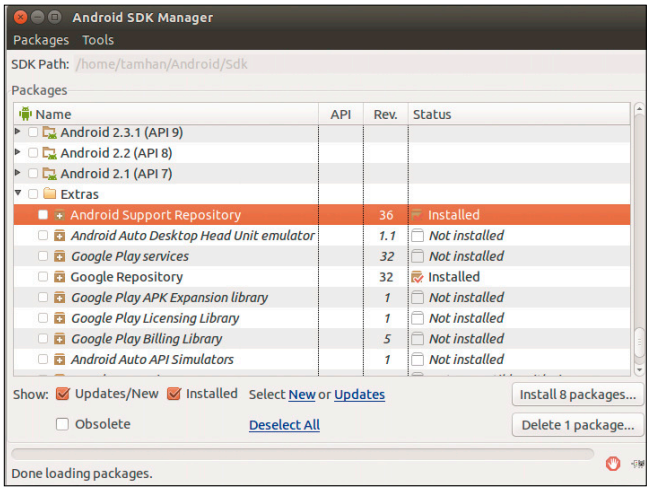
```
dependencies {
  ...
  androidTestCompile
  'com.android.support.test:runner:0.4'
  androidTestCompile
  'com.android.support.test:rules:0.4'
  androidTestCompile
  'com.android.support.test.espresso:espresso-core:2.2.1'
  androidTestCompile
  'com.android.support.test.uiautomator:
  uiautomator-v18:2.1.2'
}
```

Im Android-Block sind ebenfalls Erweiterungen erforderlich. Spezifisch müssen Sie in der Rubrik *defaultConfig* festlegen, welches Programm beziehungsweise welche Klasse für die Ausführung der Testfälle zuständig ist. Wir setzen den Wert von *testInstrumentationRunner* hier auf die von Google vorgeschlagene Klasse:

```
android {
  compileSdkVersion 24
  buildToolsVersion "23.0.3"
  defaultConfig {
    testInstrumentationRunner
    "android.support.test.runner.AndroidJUnitRunner"
    ...
```

Im Rahmen der Resynchronisation der Gradle-Dateien kommt es – je nach Kombination von SDK und Bibliotheken – zu einem Fehler, der auf einen Konflikt im Bereich der Supportbibliothek hinweist. Er äußert sich normalerweise durch eine Fehlermeldung, die in Android Studio neben der Anzeige des bekannten gelben Gradle-Warnbanners aufscheint.

Die besprochenen Methode ist nicht der allein selig machende Weg. Gradle enthält einen Konfliktmanager, der derartige Probleme automatisch zu lösen versucht. Weitere Informationen zu diesem Feature finden sich unter https://docs.gradle.org/current/dsl/org.gradle.api.artifacts.ResolutionStrategy.html. Zur Behebung dieses Problems ist etwas Trickserei erforderlich. Öffnen Sie den Terminal-Teil von Android Studio und geben Sie das Kommando *./gradlew:app:androidDependencies* ein. Wundern Sie sich nicht, wenn die IDE alle Abhängigkeiten des Projekts nochmals herunterlädt: Aufrufe aus der Konsole und Aufrufe aus dem Buildsystem arbeiten mit zwei verschiedenen Bibliotheks-Caches.



Das Android Support Repository ist für Instrumented Tests zwingend erforderlich (Bild 5)

Das passende Kommando liefert im Rahmen seiner Abarbeitung einen Fehler:

```
tamhan@tamhan-thinkpad:~/Desktop/NMGCrashTester$
./gradlew :app:dependencies
...
WARNING: Conflict with dependency 'com.android.
support:support-annotations'. Resolved versions for app
(24.0.0) and test app (23.0.1) differ...
```

Kopieren Sie die gesamte Ausgabe in einen Editor Ihrer Wahl und suchen Sie nach dem Namen der monierten Biblio- ▶

thek. Auf der Workstations des Autors kam *support-annotations* in den folgenden beiden Rubriken vor:

```
androidTestCompile -
Classpath for compiling the androidTest sources.
+--- com.android.support.test:runner:0.4 -> 0.4.1
| +--- com.android.support.test:exposed-instrumentation-
api-publish:0.4.1
|    +--- com.android.support:support-annotations:23.0.1
|    \--- junit:junit:4.12
compile - Classpath for compiling the main sources.
\--- com.android.support:appcompat-v7:24.0.0
+--- com.android.support:support-v4:24.0.0
|    \--- com.android.support:support-annotations:24.0.0
```

Achten Sie dabei darauf, die mit dem String *## Internal use, do not manually configure ##* markierten Blöcke zu ignorieren. Sie werden von Gradle zur Kompilationszeit automatisch erzeugt, Änderungen an ihnen werden permanent überschrieben. Ein Blick auf die beiden weiter oben abgedruckten Teile des Abhängigkeitsbaums zeigt, dass verschiedene Versionen der Bibliothek *android-support* eingebunden werden. Zur Lösung dieses Problems muss der *dependencies*-Block folgendermaßen angepasst werden:

```
dependencies {
  compile fileTree(dir: 'libs', include: ['*.jar'])
  testCompile 'junit:junit:4.12'
  compile 'com.android.support:appcompat-v7:24.0.0'
  androidTestCompile
  'com.android.support:appcompat-v7:24.0.0'
```

Der *dependencies*-Block ist für alle Konfigurationen des *App*-Moduls gleichermaßen relevant. Jede Abhängigkeitsdeklaration beginnt mit dem Namen der Konfiguration, für die sie vorgesehen ist. Durch unsere Modifikation sorgen wir dafür, dass die Konfiguration *androidTestCompile* dieselbe Version der Supportbibliothek verwendet, die auch für normale Kompilationen verwendet wird. Die nächste Resynchronisation des Gradle-Buildsystems wird dementsprechend ohne Fehler durchlaufen. Damit können wir uns den eigentlichen Testfällen zuwenden. Die im Rahmen des Projektskeletts angelegte Datei *ApplicationTest.java* enthält von Haus aus eine Implementierung der *ApplicationTestCase*-Klasse, die wir als Erstes folgendermaßen anpassen:

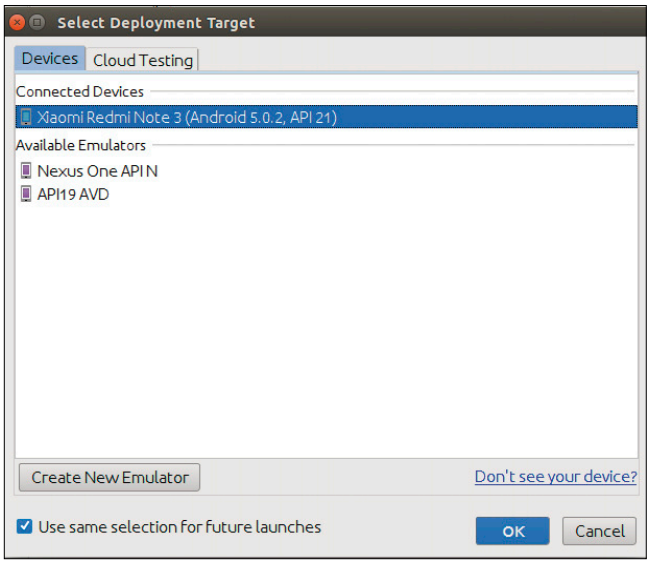
```
public class ApplicationTest extends ApplicationTestCase
<Application> {
  public ApplicationTest() {
    super(Application.class);
    Context myContext = this.getContext();
    assertNotNull(myContext);
  }
}
```

Das Ausführen von Instrumentierungstests erfolgt im Großen und Ganzen nach dem von normalen Tests bekannten Schema. Nach einem Rechtsklick auf das Projekt wählen Sie die Option *Run*. Der Unterschied zu normalen Testfällen ist, dass im Rahmen der Testausführung der bekannte Geräte-Zielauswahldialog am Bildschirm erscheint (Bild 6).

Für die Programmausführung der Testfälle sind sowohl Emulatoren als auch reale Smartphones geeignet. Wählen Sie die für Sie passende Konfiguration und starten Sie die Ausführung. Android Studio baut daraufhin eine Verbindung zum Zielsystem auf, die das Abarbeiten der Testfälle ermöglicht.

Auf den ersten Blick wirkt der weiter oben abgedruckte Testfall durchaus verlockend. Rein theoretisch müsste ja jede *Application*-Klasse einen Kontext bereitstellen, der die Assertation erfolgreich durchläuft.

Leider ist dies nicht der Fall. Wer die Testklasse ausführt, erhält eine Gruppe von Warnungen und Fehlern, die zudem irreführend sind. Wer genauer nachsieht, stellt fest, dass mehrere Exceptions geworfen werden. Eine davon weist darauf

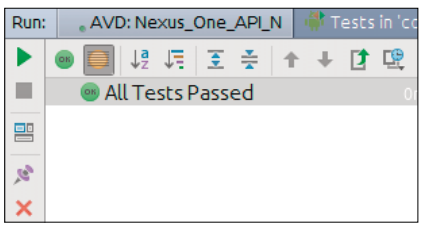


Instrumentierte Testfälle können auch auf realer Hardware ablaufen (Bild 6)

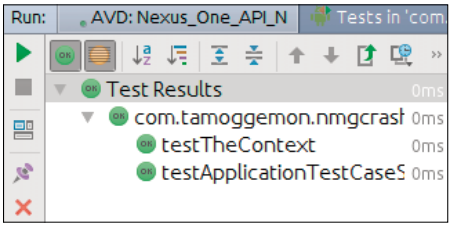
hin, dass der Userspace offensichtlich nicht richtig konfiguriert ist.

Das liegt daran, dass der Konstruktor von Testklassen keinen legitimen Testfall darstellt. Fehlt die aus dem vorigen Schritt bekannte *@Test*-Annotation, so darf die Methode keine *assert*-Calls enthalten. Eine korrekte Variante des Codes sähe folgendermaßen aus:

```
public class ApplicationTest extends ApplicationTestCase
<Application> {
  public ApplicationTest() {
    super(Application.class);
  }
  @Test
  public void testTheContext()
  {
    Context myContext = this.getContext();
```



Ist der OK-Button nicht gedrückt, so erscheinen erfolgreich abgelaufene Tests nicht in der Liste (Bild 7)



Das Anklicken von OK sorgt dafür, dass alle Testfälle gleichberechtigt im Test Runner erscheinen (Bild 8)

```
    assertNotNull(myContext);
  }
}
```

Im Benutzerinterface gibt es an dieser Stelle eine kleine Falle: Android Studio blendet erfolgreich durchgelaufene Test von Haus aus aus. Dieses Problem lässt sich durch Anklicken des grünen OK-Buttons beheben (Bild 7, Bild 8).

Um unserem Testfall zu etwas zusätzlichem Realismus zu verhelfen, beginnen wir mit der Realisierung einer für das Handhaben von Applikationspräferenzen zuständigen Klasse. Ihr Konstruktor bevölkert die interne *SharedPreference*-Instanz mit dem angelieferten Objekt:

```
public class PrefHandler {
  SharedPreferences myP;
  public PrefHandler(SharedPreferences _myP) {
    myP=_myP;
  }
```

Reale Präferenzklassen exponieren normalerweise Helfermethoden, die die eigentlichen Daten auf eine direkte Art und Weise bereitstellen. Im Fall unseres Programms sieht der dazu notwendige Code folgendermaßen aus:

```
  public void kekszahlSpeichern(int _kekse){
    SharedPreferences.Editor anEditor=myP.edit();
    anEditor.putInt("Kekse", _kekse);
    anEditor.apply();
  }
  public int kekszahlLesen()  {
    return myP.getInt("Kekse", -1);
  }
```

Damit können wir uns der Realisierung des eigentlichen Testfalls zuwenden. Er beschafft im ersten Schritt eine *SharedPreferences*-Instanz, die sodann zur Realisierung der Helferklasse eingespannt wird. Diese wird zu guter Letzt mit einigen Tests befüllt, die das korrekte Funktionieren des Zahlspeichers bestätigen:

```
@Test
public void testPrefHandling() {
  PrefHandler newHandler=new PrefHandler(getContext().
  getSharedPreferences("Testprefs",0));
  newHandler.
  kekszahlSpeichern(22);
  assertEquals(newHandler.
  kekszahlLesen(),22);
}
```

Führen Sie den Testfall bei Interesse aus, um sich am erfolgreichen Funktionieren instrumentierter Testfälle zu erfreuen. Dank der Ausführung in der realen Android-Runtime können Sie hier auch auf statische Methoden zurückgreifen; das manuelle Erzeugen von Code auf Basis von Mockito entfällt. Wer seine Unit-Tests in einem Emulator ausführt, sollte darauf achten, ein x86-Image zu verwenden. Die Workstation ist dabei von der Emulation einer fremden Prozessorarchitektur befreit, was sich in Form höherer Rechenleistung auswirkt.

Und jetzt auf GUI-Ebene

Unit-Tests sind unter Entwicklern von GUI-getriebenen Applikationen verschrien. Wenn der Gutteil der Fehler als im GUI entstehend empfunden wird, wirkt das Testen der Logik wie eine wenig sinnvolle Fleißaufgabe. In der Vergangenheit löste man dieses Problem gern durch einen Test-Kriterienkatalog, der von Testern manuell abgearbeitet werden musste.

Google löst das Problem durch Einbindung des Espresso-Frameworks, das wir weiter oben im Rahmen der Einrichtung instrumentierter Tests in unser Projekt integriert haben. Daher können wir an dieser Stelle einfach draufloscodieren.

Leider gibt es eine kleine Besonderheit zu beachten: Die in Espresso implementierte Zugriffslogik kommt mit Animationen nicht sonderlich gut zurecht. Wechseln Sie in die Rubrik *Settings, Developing Options* und deaktivieren Sie die folgenden drei Optionen: *Window animation scale, Transition animation scale* und *Animator duration scale*.

Im nächsten Schritt legen wir eine neue Datei namens *ActivityTestClass.java* an. Sie wird mit folgendem Inhalt ausgestattet:

```
@RunWith(AndroidJUnit4.class)
public class ActivityTestClass {
  @Rule
  public ActivityTestRule<MainActivity> mActivityRule
  = new ActivityTestRule<>(MainActivity.class);
```

Die *RunWith*-Notation informiert die Ausführungsumgebung darüber, welche Klasse für die Ausführung des Testfalls zuständig ist. Das *Rule*-Attribut dient zur Festlegung von Bedingungen zur Testausführung. Stellen Sie es sich als eine Art globale Variable vor. Wir nutzen hier eine Instanz der Klasse *ActivityTestRule*, die als Wrapper um eine zu überprüfende Activity dient.

Der eigentliche Test erfolgt sodann – wie üblich – in einer als *Test* markierten Funktion. Die *onView*-Methode ist ein von Espresso bereitgestellte Hilfselement, das das Suchen von Objekten in der gerade geöffneten Activity ermöglicht. ▶

Die Funktion übernimmt ein als Matcher bezeichnetes Objekt, das die eigentliche Suchlogik bereitstellt. Im nächsten Schritt folgt ein Aufruf von *check:* Diese Funktion übernimmt ebenfalls ein Objekt, welches das zu prüfende Kriterium beschreibt:

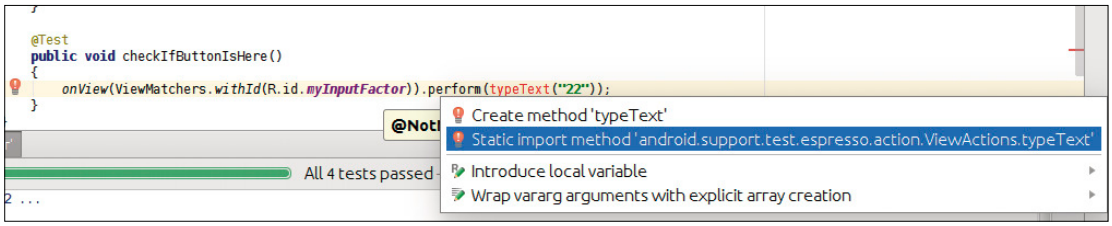
```
  @Test
  public void checkIfButtonIsHere() {
    onView(ViewMatchers.withText
    ("Fehlender Text")).check(doesNotExist());
  }
}
```

Im Fall unseres Objekts ist die Situation einfach. Da es keinen Knopf mit dem betreffenden Text im Formular gibt, läuft der Testfall erfolgreich durch.

Espresso definiert die einzelnen Member-Elemente als statische Methoden. Aus diesem Grund stellt sich die in Android Studio integrierte IntelliSense-Logik mitunter etwas quer: Wer das per [Alt Eingabe] aufrufbare Kontextmenü auf den Bildschirm holt, muss die in Bild 9 gezeigte Option wählen.

Im Internet finden sich immer wieder Verweise auf die Klasse *ActivityInstrumentationTestCase2*. Sie wurde von Google im API-Level 24 abgekündigt und sollte in neuen Projekten nicht mehr verwendet werden.

Espresso ist nicht darauf beschränkt, das Vorhandensein von Steuerelementen zu prüfen. Als nächste Amtshandlung



Die Option Static import bindet die fehlenden Elemente ins Hauptprojekt ein (Bild 9)

wollen wir unser Formular um drei Widgets erweitern, die eine grundlegende Berechnungsoperation realisieren:

```
<Button
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Kalkulieren"
  android:id="@+id/buttonCalc"
  android:layout_below="@+id/myOutField"
  android:layout_alignParentStart="true" />
<EditText
  android:layout_width="fill_parent"
  android:layout_height="60dp"
  android:id="@+id/myInputFactor"
  android:layout_below="@+id/buttonCrash"
  android:layout_centerHorizontal="true" />
<EditText
  android:layout_width="fill_parent"
  android:layout_height="60dp"
  android:id="@+id/myOutField"
  android:layout_below="@+id/myInputFactor"
  android:layout_centerHorizontal="true" />
```

Auch die Implementierung ist nicht sonderlich komplex. Es handelt sich hier im Großen und Ganzen um Standard-Android-Code, der das Potenzieren einer Zahl realisiert. Um die Prüfung des Verhaltens zu ermöglichen, wandert das Ergebnis sodann in das als Ausgabeparameter designierte Feld:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  ...
  findViewById(R.id.buttonCalc).
  setOnClickListener(this);
}
@Override
public void onClick(View view) {
  if(view==findViewById(R.id.buttonCrash)){throw new
  NullPointerException();}
  else {
    EditText input=(EditText)findViewById
    (R.id.myInputFactor);
    EditText output=(EditText)findViewById
    (R.id.myOutField);
    Float outVal=Float.parseFloat(input.getText().
    toString()) * Float.parseFloat(input.getText().
    toString());
    output.setText(outVal.toString());
  }
}
```

Realisieren wir nun den zweiten Testfall:

```
@Test
public void checkIfLogicWorks() {
  onView(ViewMatchers.withId(R.id.myInputFactor)).
  perform(typeText("2"));
  onView(ViewMatchers.withId(R.id.buttonCalc)).
  perform(click());
  onView(ViewMatchers.withId(R.id.myOutField)).
  check(matches(withText("4.0")));
}
```

Er beginnt mit dem Eingeben eines Wertes in das Eingabefeld, wozu der *typeText*-Operator erforderlich ist. Im nächs-

ten Schritt wird abermals die *perform*-Methode aufgerufen, die nun für das Anklicken des Berechnungsbuttons zuständig ist. An dieser Stelle zeigt sich die Syntax von Espresso abermals: *perform* nimmt ein Objekt entgegen, das die in der Benutzerschnittstelle durchzuführende Handlung beschreibt. Während in der ersten Operation *typeText* samt Parameter aktiviert wurde, folgt nun ein Aufruf von *click()*.

Verifikation

Damit fehlt nur noch die Verifikation, die über das im vorigen Abschnitt erstmals besprochene *check*-Kommando erfolgt. Dieses Mal übergeben wir einen Matcher, der den Inhalt des *Text*-Attributs des jeweiligen Steuerelements analysiert und uns so Rückschlüsse auf das korrekte Funktionieren unserer Berechnungslogik ermöglicht.

Wer den Testfall in einem Emulator ausführt, wird manchmal mit einer irreführenden Fehlermeldung der Bauart *Error performing 'single click' on view 'with id: com.tamoggemon.nmgcrashtester:id/buttonCalc'*. konfrontiert. Dies ist insofern lästig, als der Fehler auf eine SecurityException zurückgeht, die sich erst weiter unten im Stack finden lässt.

Als erste Maßnahme zur Behebung derartiger Fehler bietet sich die Aktualisierung der verwendeten Version von Espresso an. Das Entwicklerteam hat vor einiger Zeit ein stilles Update durchgeführt, das in der Hauptdokumentation noch nicht reflektiert wurde. Öffnen Sie die *.gradle*-Datei des *App*-Moduls und passen Sie die Inklusion von Espresso folgendermaßen an:

```
dependencies {
  ...
  androidTestCompile
  'com.android.support.test.espresso:espresso-core:2.2.2'
```

Funktioniert auch dies nicht (Resynchronisation der Build-Umgebung beachten), so bietet sich nur die Auswanderung auf ein Gerät an. Der Autor testete das Programm mit einem unter Android 6.0 laufenden Prototyp des Apollo Vernee Lite: Die Abarbeitung des Testfalls erfolgte ohne jegliches Problem.

Direkte Attacke

Das Arbeiten mit den in Espresso implementierten Funktionen und Matchern ist nicht jedermanns Sache. Es handelt sich dabei immerhin um eine komplett neue Syntax, die man sich erst einmal aneignen muss. Espresso-Tests laufen im Rahmen des Betriebssystems ab.

Aus diesem Grund spricht nichts dagegen, das normale Android-API zur Realisierung von Interaktionen und zur Verifikation der Resultate einzusetzen. Ein schönes Beispiel dafür wäre folgender Testfall, der das Vorhandensein eines Steuerelements prüft:

```
@Test
public void directCoding() {
  MainActivity anActivity=mActivityRule.getActivity();
  EditText myView=(EditText)anActivity.findViewById
  (R.id.myInputFactor);
  Assert.assertNotNull(myView);
}
```

Der erste Unterschied zu gewöhnlichem Android-Code ist die Beschaffung der *Activity*-Instanz: Sie erfolgt nun durch einen Aufruf der Methode *getActivity()*, die in der vom *@Rule*-Statement bereitgestellten Klasse auf Aktivierung wartet.

Der zweite Unterschied – insbesondere im Vergleich zu den weiter oben abgedruckten Testfällen – ist die Art des Aufrufs von *Assert*. Die diversen zur Teststeuerung vorgesehenen Methoden werden nun über ihr statisches Mutterobjekt aufgerufen. Wer dies nicht möchte, kann sie auch über ein *Include*-Statement einbinden.

Sonst gibt es an dieser Stelle nur wenig Wichtiges. Behalten Sie im Hinterkopf, dass die Abarbeitung der Testfälle nicht festgelegt ist. Gehen Sie in der Praxis davon aus, dass die Activity immer neu gestartet wird – mehrfache Aufrufe des Konstruktors gehören dazu.

Zu Espresso findet man in der offiziellen Google-Dokumentation nur wenig. Das liegt daran, dass die diesbezüglichen Informationen auf den URL https://google.github.io/android-testing-support-library/docs/espresso ausgelagert wurden.

Automatisierte Fehlersuche

Hundertprozentige Testabdeckung ist in realen Programmen nur sehr selten mit sinnvollem Aufwand erreichbar. Es gibt immer irgendetwas, was unter den sprichwörtlichen Teppich fällt. Da Rechenleistung preiswert ist, bietet sich die zufällige Suche nach Fehlern an.

Die dahinterstehende Idee ist nicht neu: Palm bot seinen Entwicklern mit den Gremlins erstmals ein automatisiertes Testsystem an, das eine oder mehrere Applikationen mit zufällig generierten Klick- und Eingabeereignissen zuschüttete. Die Android Debug Bridge enthält mit Monkey ein ähnliches Werkzeug.

Zur Nutzung von Monkey ist wenig Zusatzaufwand erforderlich. Installieren Sie die zu testende Applikation auf ein Smartphone oder einen Emulator, und setzen Sie per ADB den folgenden Befehl ab:

```
$ adb shell monkey -p your.package.name -v 500
```

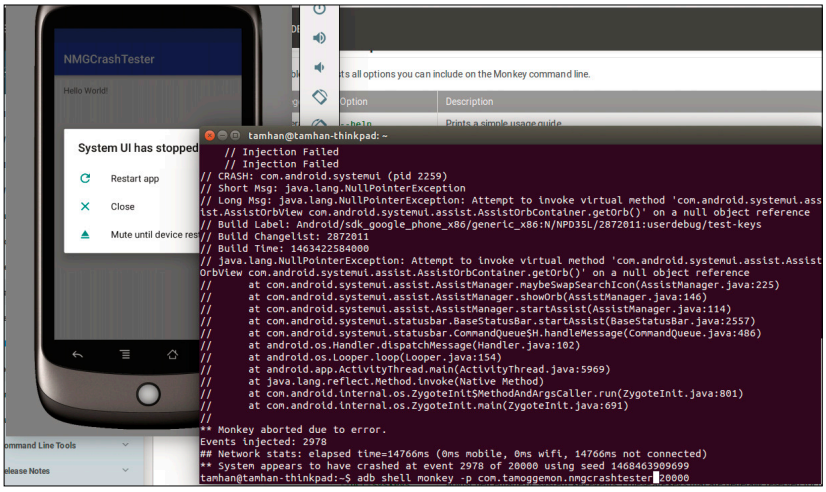
Dabei legt *v* die Menge der abzufeuernden Ereignisse fest – in der Praxis sorgen einige Tausend Events auch auf leistungsstarken Workstations für mehr als ausreichende Rechenlast.

Wer das Konsolenfenster nicht jedes Mal in das SDK-Verzeichnis navigieren möchte, kann den Pfad mit ADB und Co. auch zur *PATH*-Variable seiner Workstation hinzufügen.

Witzigerweise finden Monkey-Testläufe insbesondere auf nicht finalisierten Emulatoren immer wieder Fehler im Betriebssystem. Bild 10 zeigt ein Beispiel für einen Crash, für den die zu testende Applikation nicht verantwortlich ist. Für Sie als testwilliger Programmierer ist dies insofern ärgerlich, als der Testlauf an dieser Stelle abbricht und von Hand neu gestartet werden muss. ▶

Monkey hat wegen der Zufälligkeit der Eingaben Probleme damit, längere Prozesse erfolgreich abzuarbeiten. Entwickler können hier durch spezielle Monkey-Einsprungpunkte Abhilfe schaffen – es handelt sich dabei um für Monkey dedizierte Activities, die in der Manifestdatei folgendermaßen angelegt werden:

```
<manifest ...>
  <application ...>
    ...
    <activity android:name=
    ".MonkeyActivity">
      <intent-filter>
        <action android:name=
        "android.intent.action.MAIN" />
        <category android:name=
        "android.intent.category.MONKEY" />
      </intent-filter>
    </activity>
```



Hier ist der Entwickler unschuldig … (Bild 10)

Insbesondere bei Applikationen mit einem Installationsassistenten ist es ratsam, dem Affen einen direkten Weg ins Programm zu bieten. Achten Sie lediglich darauf, dass die dabei voreingestellten Werte realistisch sind.

Wer Python beherrscht, kann die Arbeit mit Monkey über das unter https://developer.android.com/studio/test/monkeyrunner beschriebene MonkeyRunner-Werkzeug automatisieren.

Meide die Settings

In der Praxis nervt Monkey – zumindest auf der Workstation des Autors – mit einer gewissen Affinität zu Einstellungs- und sonstigen Dialogen, die sich auch durch Nutzung des Parameters *-p* nicht wesentlich einschränken lässt. Der von Diego Torres Milano entwickelte und unter https://github.com/dtmilano/AndroidViewClient zum Download bereitstehende AndroidViewClient schafft mit manueller Intelligenz Abhilfe. Es handelt sich dabei um ein Programmierframework, in dem der Entwickler Testabläufe durchautomatisieren kann.

Zum Herunterladen des Produkts ist ein Python-Interpreter erforderlich. Die folgenden Schritte beschreiben das Deployment unter Ubuntu 14.04 LTS. Nutzer anderer Betriebssysteme finden unter https://github.com/dtmilano/AndroidViewClient/wiki Rat und Hilfe. Öffnen Sie ein Terminal und geben Sie folgende Kommandos ein:

```
tamhan@tamhan-thinkpad:~$ sudo apt-get install
python-setuptools
...
tamhan@tamhan-thinkpad:~$ sudo easy_install
--upgrade androidviewclient
```

Laden Sie sodann das unter https://github.com/dtmilano/AndroidViewClient/archive/master.zip verfügbare Codearchiv herunter und entpacken Sie seinen Inhalt an einen gut zugänglichen Platz im Dateisystem. Bei erfolgreicher Installation verhält sich das Skript *check-import.py* folgendermaßen:

```
tamhan@tamhan-thinkpad:
~/AndroidViewClient-master/examples$ python
check-import.py
```

AndroidViewClient tut selbst nicht sonderlich viel. Nützlich ist das Framework nur dann, wenn man es mit handgeschriebener Logik ausstattet. Ein Beispiel dafür wäre das Aktivieren eines am Bildschirm befindlichen Buttons – ergänzen Sie *MainActivity* dazu um einen Button, dessen Event Handler – im Idealfall durch das Werfen eines Objekts – eine Exception verursacht:

```
public class MainActivity extends AppCompatActivity
implements View.OnClickListener {
  @Override

  protected void onCreate(Bundle
```

```
tamhan@tamhan-thinkpad: ~/Desktop/Stuff/AndroidViewClient-master/examples
tamhan@tamhan-thinkpad:~/Desktop/Stuff/AndroidViewClient-master/examples$ dump
/usr/local/lib/python2.7/dist-packages/androidviewclient-11.5.11-py2.7.egg/com/dtmilano/android/viewclient.py:2601: UserWarning:
 Running on emulator but no serial number was specified then 'emulator-5554' is used
  warnings.warn("Running on emulator but no serial number was specified then 'emulator-5554' is used")
android.widget.FrameLayout
   android.widget.TextView  NMGCrashTester
   android.widget.TextView com.tamoggemon.nmgcrashtester:id/textView Hello World!
   android.widget.Button com.tamoggemon.nmgcrashtester:id/buttonCrash New Button
   android.widget.EditText com.tamoggemon.nmgcrashtester:id/myInputFactor
   android.widget.EditText com.tamoggemon.nmgcrashtester:id/myOutField
   android.widget.Button com.tamoggemon.nmgcrashtester:id/buttonCalc Kalkulieren
tamhan@tamhan-thinkpad:~/Desktop/Stuff/AndroidViewClient-master/examples$
```

Dump liefert Informationen über die am Bildschirm befindlichen Views zurück (Bild 11)

```
  savedInstanceState) {
    ...
    findViewById(R.id.buttonCrash).
    setOnClickListener(this);
  }
  @Override
  public void onClick(View view) {
    throw new NullPointerException();
  }
}
```

Schicken Sie das Programm im nächsten Schritt in einen Emulator oder auf ein reales Endgerät: AndroidViewClient enthält keine eigene Auslieferunglogik und ergreift das zum Ausführen der Testfälle vorgesehene Gerät per ADB.

Als erster Versuch bietet sich das Auswerfen der Komplettkonfiguration an. Es lässt sich durch Eingabe des Befehls *dump* bewerkstelligen. Bild 11 zeigt, was Sie sich erwarten dürfen.

Kommt es stattdessen zu einem Fehler der Bauart *ValueError: received does not contain valid XML: /system/bin/sh: cat: /storage/self/window_dump.xml: No such file or directory*, so haben Sie es wahrscheinlich mit einem fehlerhaft konfigurierten Emulator zu tun. Während der Versuche des Autors funktionierte das Framework mit Android N wegen einer Änderung in der Organisation des Festwertspeichers nicht. Die in Bild 12 gezeigte Konfiguration funktionierte hingegen problemlos.

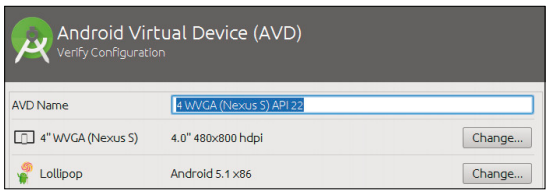
Als Nächstes wollen wir den Button programmatisch aktivieren. Dazu adaptieren wir ein auf *click-button-by-text.py* aufgebautes Skript. Legen Sie eine neue Datei namens *click-nmg.py* an, und statten Sie sie mit folgendem Code aus:

```
#! /usr/bin/env python
import sys
import os
import time
```

Nach dem unter Unix zur Kennzeichnung der Ausführungsumgebung obligaten Hashbang-Zeichen folgt das Importieren dreier Unterstützungsbibliotheken aus der Python-Runtime. Im nächsten Schritt wird das Heimverzeichnis des AndroidViewClients zum Pfad hinzugefügt. Scheitert dies, so wird die Skript-Abarbeitung trotzdem weitergeführt:

```
try:
  sys.path.append(os.path.join(os.environ['ANDROID_VIEW_
  CLIENT_HOME'], 'src'))
except:
  pass

from com.dtmilano.android.viewclient import ViewClient
```

Das Erzeugen des *vc*-Objekts sorgt dafür, dass der AndroidViewClient Kontakt mit dem Zielgerät aufnehmen kann. Die von ihm exponierte Methode *findViewWithText* sucht nach einem Steuerelement, das den angegebenen Text enthält:



Der Emulator von Android 5.1 ist für den AndroidViewClient geeignet (Bild 12)

```
vc = ViewClient(*ViewClient.connectToDeviceOrExit())

bt='New Button'
b = vc.findViewWithText(bt)
```

Wenn der angezeigte Text zum Finden eines Knopfes geeignet war, so beschafft das Skript im nächsten Schritt die Koordinaten des Knopfes. Der Aufruf von *b.touch()* setzt dann ein Klickereignis ab. Die Koordinatenbeschaffung erfolgt hier nur aus didaktischem Interesse:

```
if b:
  (x, y) = b.getXY()
  print >>sys.stderr, "clicking b%s @ (%d,%d) ..." %
  (bt, x, y)
  b.touch()
else:
  print >>sys.stderr, "b%s not found" % bt
```

Vor der eigentlichen Programmausführung ist unter Unix noch das Setzen des Executable-Bits nowendig, was durch Eingabe des Befehls *chmod +x click-nmg.py* bewerkstelligt werden kann.

Nach getaner Arbeit folgt die Ausführung, die die im Emulator laufende Applikation mit einer Unhandled-Exception terminiert. An dieser Stelle sind Sie frei. Das Python-API des Produkts lässt sich zur Realisierung beliebiger Interaktionsszenarien einspannen. So wäre die Kombination eines Zufallsgenerators mit dem *touch*-Befehl denkbar. Alternativ dazu könnten Sie auch die von *dump* angelieferten Ergebnisse auswerten, um das Klicken nur auf aktive Steuerelemente zu beschränken.

Fazit

Nach Ansicht des Autors gilt der olympische Gedanke auch beim Unit Testing: Jeder durch einen automatisierten Testfall vor der Programmauslieferung gefundene Fehler ist einer, der die Bewertungen der App im Store nicht schädigt. ■

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

VERSIONSVERWALTUNG MIT GIT IM FREIEN ONLINE-REPOSITORY GITHUB

Versionierte Cloud für Projektdateien

Ganz gleich ob Quellcode, Grafikdatei oder Dokumentation – Arbeitsfortschritte gehören versioniert, um später auf ältere Versionsstände zurückgreifen zu können, am besten direkt online mit GitHub.

Viele Open-Source-Projekte setzen inzwischen auf das verteilte Versionskontrollsystem Git und auf das Online-Repository-System GitHub. Für öffentlich sichtbare Projekte und Open-Source-Projekte ist GitHub kostenlos, und deswegen ist das System besonders bei Open-Source-Entwicklern sehr beliebt. Wer sein Projekt nicht der Öffentlichkeit zugänglich machen möchte, kann zu einem kostenpflichtigen Account wechseln; hierfür fallen mindestens sieben US-Dollar im Monat an. In diesem Artikel wird ausschließlich die kostenlose Variante genutzt.

Git ist dabei mehr als ein Online-Repository, es ist vielmehr eine Online-Community für Projektdateien. Wer Open-Source-Projekte nutzt, sollte auch GitHub bedienen können. GitHub nutzt im Hintergrund Git, deshalb soll hier zunächst in die Nutzung von Git eingeführt werden. Git selbst ist unabhängig von GitHub und kann daher auch ohne das Online-Repository genutzt werden.

Grundlagen von Git

Die beliebten Tastaturkürzel [Strg Z] für Undo und [Strg Y] für Redo kennt und nutzt sicher jeder. Nur irgendwann ist der Zwischenspeicher des Betriebssystems erschöpft, spätestens nach einem Reboot des Systems.

Genau hier kommt ein Versionskontrollsystem (Version Control System, VCS) ins Spiel. Änderungen, auch von mehreren Personen, werden dort gespeichert und können wiederhergestellt werden. Ein sehr beliebtes System im Open-Source-Bereich ist Git. Bevor man nun aber mit Git arbeitet, sollte man die Grundkonzepte dieses VCS verstehen. Entwickler, die von einem zentralen System wie Subversion zu Git wechseln, sollten sich auch zuerst mit den Grundlagen von Git beschäftigen, denn die Unterschiede zu einem zentralen Versionierungssystem sind teilweise erheblich.

Aufbau des Beispielprojekts (Bild 1)

Git installieren

Git steht für die Betriebssysteme Linux, Mac OS X, Solaris und Windows zur Verfügung. Bei aktuellen Mac-OS-X-Versionen ist Git bereits installiert, deshalb ist hier keine manuelle Installation erforderlich. Für Windows muss eine EXE-Datei heruntergeladen werden. Nach dem Download der aktuellen Version 2.10.0 (Stand September 2016) muss die EXE-Datei per Doppelklick gestartet werden, und nach kurzer Zeit steht Git in der Kommandozeile von Windows zur Verfügung.

Ein großer Unterschied zu Systemen wie CVS (Concurrent Versions Systems) und Subversion ist die Art, wie Dateien im VCS verwaltet werden. Bei Subversion werden die abgelegten Informationen als eine Liste von fortlaufenden Änderungen gespeichert. Bei Git werden immer komplette Stände abgelegt. Damit keine doppelte Datenhaltung auftritt, werden gleiche Dateien über eine Referenz verwaltet. Das bedeutet, falls sich eine Datei über mehrere Versionen nicht ändert, ist in einer spezifischen Repository-Version jeweils immer nur eine Referenz auf die Ursprungsdatei gespeichert.

Drei Bestandteile

Ein Repository besteht grundsätzlich aus drei Bestandteilen: Dateien, Verzeichnisse und Versionen. Dateien kommen in Form von Texten oder binären Daten vor. Die Dateien werden auch als Blobs bezeichnet. Die Daten werden unabhängig vom Dateinamen gespeichert. Dann gibt es Verzeichnisse, die Inhalte – also die Blobs – mit Dateinamen verknüpfen. Verzeichnisse werden auch als Trees bezeichnet. Versionen stellen einen wiederherstellbaren Zustand eines Verzeichnisses, inklusive seiner Unterverzeichnisse, dar. Eine Version wird in Git als Commit bezeichnet. Der Commit wird über einen eindeutigen Hashwert repräsentiert.

Ein Git-Projekt besteht aus mehreren Verzeichnisinstanzen. Die erste Instanz ist das lokale Arbeitsverzeichnis mit den Projektdateien. Danach folgt der Index, auch Stage genannt. Die letzte Stufe ist das lokale Repository. Der zuletzt ausgeführte Commit auf einem Repository wird dabei auch

als HEAD bezeichnet. Alle drei Verzeichnisinstanzen (Arbeitsverzeichnis, Index und HEAD) liegen auf dem lokalen Rechner. Daneben existiert dann noch ein entferntes Repository. Der Hauptzweig in diesem Repository wird als master bezeichnet.

Lernen am Beispiel

An einem Beispielprojekt soll die grundlegende Arbeit mit dem VCS Git gezeigt werden. Das Beispielprojekt (Bild 1) hat ein Hauptverzeichnis mit drei Dateien *(text1.txt, text2.txt* und *text3.txt)* und ein Unterverzeichnis *(sub)* mit weiteren zwei Dateien *(sub_text1.txt* und *sub_text2.txt)*.

Um für das Projekt ein Repository anzulegen, muss im Hauptverzeichnis der Befehl *git init* auf der Kommandozeile ausgeführt werden. Dieser Befehl erzeugt ein Unterverzeichnis *.git*. In diesem Unterverzeichnis liegen das Git-Repository und dessen Metadaten für das Projekt. Nach der Initialisierung des Projekts über den Befehlt *get init* befinden sich noch keine Daten im Repository. Um Dateien hinzuzufügen, sind zwei Schritte notwendig.

Zuerst werden Dateien über *git add* dem lokalen Index hinzugefügt. Entweder gibt man bei diesem Befehl einzelne Dateinamen an, zum Beispiel *git add text1.txt*, oder man fügt mit *git add ** alle Dateien inklusive der Unterverzeichnisse zum lokalen Index hinzu. Nun müssen die Änderungen noch mit *git commit* bestätigt werden, erst danach befinden sich die Dateien im Repository. Bei der Ausführung von *git commit* muss noch ein Kommentar über den Parameter *-m* (Abkürzung für *message)* mitgegeben werden. Danach wird dieser Commit zum HEAD:

```
$ git add *
$ commit -m "Erste Version des Projekts"

[master (root-commit) f5d7f1f] Erste Version des
Projekts
5 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 sub/sub_text1.txt
create mode 100644 sub/sub_text2.txt
create mode 100644 text1.txt
create mode 100644 text2.txt
create mode 100644 text3.txt
```

Jeder Commit hat einen Hashwert, im Beispiel ist dieser f5d-7f1f. Über diesen Wert lässt sich der Commit eindeutig im Repository identifizieren. Der Hashwert hat eine Länge von 40 Zeichen, kann aber auch verkürzt angegeben werden. Zusätzlich wird nach *create mode* angegeben, als welcher Typ das Objekt angelegt wird. Im obigen Beispiel wurden fünf nicht ausführbare Dateien (100644) angelegt. Ein Verzeichnis hat den Wert 040000 und eine ausführbare Datei den Wert 100755. Über den Commit-Hash lässt sich der Inhalt eines Commits über den Befehl *git ls-tree* anzeigen. Es muss dabei nicht der komplette Commit-Hash angegeben werden, sondern es reichen die ersten eindeutigen Zeichen.

Die durch den Befehl *git add* durchgeführten Änderungen werden zunächst im lokalen Index (Stage) abgelegt. Diese

Listing 1: Status des Arbeitsverzeichnisses überprüfen

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

  modified:   text1.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be
  committed)
  (use "git checkout -- <file>..." to discard
  changes in working directory)

  modified:   text2.txt
```

Änderungen können über *git reset* rückgängig gemacht werden. Bei der Ausführung des Befehls *git reset* werden die letzten Änderungen aus dem Index gelöscht und der Index wird auf den HEAD zurückgesetzt. Es kann aber auch nur eine einzelne Änderung aus dem Index genommen werden.

Zum Beispiel wird über das Kommando *git reset text1.txt* nur die Änderung an der Datei *text1.txt* aus dem Index genommen. Möchte man die lokalen Änderungen im Arbeitsverzeichnis rückgängig machen, so kann man den Index über *git checkout* ins Arbeitsverzeichnis kopieren.

Auch hier kann nur eine einzelne Datei aus dem Index kopiert werden; hierfür muss der Dateiname noch als zusätzlicher Parameter angegeben werden. Wenn man bei *git checkout* zusätzlich den Hashwert eines Commits angibt, so kann man die Daten aus dem angegebenen Commit wiederherstellen.

Wenn man länger nicht mit einem Projekt gearbeitet hat oder viele Änderungen durchgeführt hat, bietet es sich an, den Status des aktuellen Arbeitsverzeichnisses über das Kommando *git status* zu überprüfen. Hierüber wird zum Beispiel angezeigt, ob Dateien zum Commit anstehen oder ob Änderungen für die Übertragung in den Index vorhanden sind.

In Listing 1 wurde die Datei *text1.txt* geändert und schon über *git add* im Index hinzugefügt, sie ist aber noch nicht im Repository. Die Datei *text2.txt* wurde geändert, wurde aber noch nicht über *git add* in den Index übertragen.

Ein Repository besitzt eine Logdatei, die man sich über den Befehl *git log* ansehen kann. Eine verkürzte Ausgabe erhält man über den Befehl *git log --oneline*.

Mit anderen Entwicklern zusammenarbeiten

Um mit mehreren Entwicklern an einem Git-Projekt zu arbeiten, müssen die Daten untereinander ausgetauscht werden. Bevor man nun mit einem Server-Repository arbeitet, sollte man erst eine Trockenübung für das Kennenlernen der wichtigsten Befehle machen. Hierfür erzeugt man sich zunächst einen Klon des vorhandenen Repositorys in ein zweites ▶

Benutzer und E-Mail

Zu einem Commit gehört immer ein Benutzer mit E-Mail-Adresse, dieser ist in der globalen Konfigurationsdatei von Git hinterlegt. Sollte man bisher keinen Benutzernamen und keine E-Mail-Adresse festgelegt haben, so erscheint eine Warnmeldung, dass beide Werte vom System auf Basis des lokalen Anmeldenamens erzeugt werden. Um die Meldung zu unterbinden, muss der Name in der globalen Konfigurationsdatei eingetragen werden. Über den Befehl *git config --global --edit* öffnet sich die Datei und die Werte können eingetragen werden:

```
$ git config --global --edit

# This is Git's per-user configuration file.
[user]
name = staeuble
email = staeuble@localhost
```

Alternativ können die Werte auch direkt über die Kommandozeile gesetzt werden:

```
git config --global user.name "Your Name"
git config --global user.email you@example.com
```

Bei der nächsten Ausführung von *git commit* werden nun diese gesetzten Werte herangezogen und es erscheint keine Warnung mehr.

Verzeichnis. Ein Klon wird über den Befehl *git clone* erzeugt, in diesem Beispiel *git clone git-sample git-sample-copy*:

```
$ git clone git-sample/ git-sample-copy
Cloning into 'git-sample-copy'...
done.
```

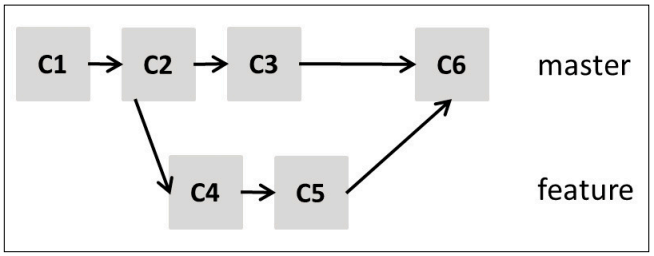
Über den Befehlt *git clone* können Sie sich somit jedes Git-Projekt auf Ihren lokalen Rechner laden. Anstelle eines lokalen Pfades können Sie nämlich auch einen URL angeben. Sollte man bereits ein geklontes Projekt vorliegen haben, so kann man sich den letzten Commit *(HEAD)* über *git pull* abholen. Eine zusätzliche Pfadangabe zum Quellrepository ist nicht nötig, diese wurde am ursprünglichen Klon gespeichert. Listing 2 zeigt dies am Beispiel. Zuerst werden zwei Dateien im Originalrepository geändert und per Commit hinzugefügt, der entsprechende Hashwert ist 362f61f. Über *git pull* werden die Änderungen im Klon abgeholt, lokal gespeichert und als neuer Commit angelegt. Dies funktioniert nur, solange beim Merge keine Konflikte auftreten. Sollten Konflikte auftreten, müssen diese manuell gelöst werden.

Der Befehl *git pull* kann auch mit Parameter genutzt werden, darüber kann dann jedes beliebige Projekte geladen werden. Wichtig ist nur, dass das aktuelle Projekt ein Git-Repository besitzt. Sollte dies nicht der Fall sein, so muss dies über *git init* angelegt werden oder das Repository muss über *git clone* kopiert werden. Beim *pull*-Befehl kann auch der Name eines Entwicklungszweigs (Branch) angegeben werden.

Tags vergeben

Wie schon erwähnt, wird ein Commit über einen 40-stelligen Hashwert eindeutig identifiziert. Andere Versionierungssysteme arbeiten mit Versionsnummern oder Bezeichnungen. Auch in Git können Bezeichnungen für einen Commit vergeben werden, diese werden in Git als Tag bezeichnet. Ein Tag wird über den Parameter *tag,* gefolgt vom Namen des Tags, vergeben. Um dem letzten Commit eine Bezeichnung zu geben, muss der Befehl *git tag 1.00* abgesetzt werden. Damit auch andere Entwickler mit einer Bezeichnung etwas anfangen können, kann man zusätzlich einen Kommentar zu einem Tag setzen: *git tag 1.00 –m "Integrationsversion"*. Sollte man versehentlich ein Tag gesetzt haben, so kann dieses über den Parameter *-d* gelöscht werden. Das zuvor gesetzte Tag wird somit über *git tag -d 1.00* gelöscht.

Nun hat der letzte Commit die Bezeichnung *1.00.* Alle Tags eines Git-Repositorys kann man sich über den Befehl *git tag*



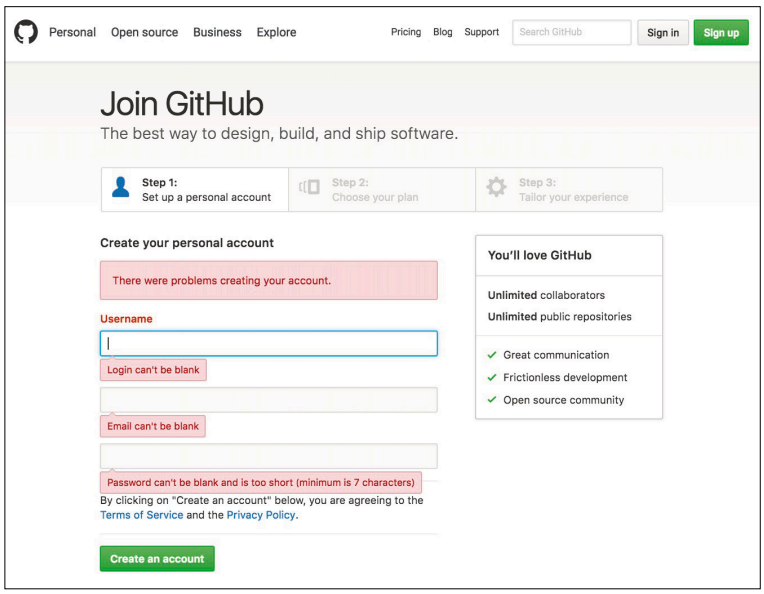
Der Branch *feature* wird mit *master* vereinigt (Bild 2)

Listing 2: Objekte des Commits f5d7f1f, inklusive der Inhalte der Verzeichnisse

```
git-sample mstaeuble$ git commit -m "test"
[master 362f61f] test
  2 files changed, 2 insertions(+), 1 deletion(-)

git-sample-copy mstaeuble$ git pull
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
From /Users/mstaeuble/projects/git-sample
  f7caf10..362f61f  master     -> origin/master
Updating f7caf10..362f61f
Fast-forward
  text1.txt | 2 +-
  text2.txt | 1 +
  2 files changed, 2 insertions(+), 1 deletion(-)
```

anzeigen lassen. Möchte man zusätzlich die vergebenen Kommentare zu einem Tag sehen, so muss man das Kommando *git tag –n* absetzen. Bei einem Projekt mit vielen Tags kann man auch gezielt nach Tags suchen. Zum Beispiel werden mit *git tag 1** alle Tags angezeigt, die mit *1* beginnen. Bei der Übertragung von Änderungen in ein anderes Repository über *git push* werden die Tags nicht übertragen. Wird der Name des Tags explizit angegeben, so wird dieser auch übertragen. Über *push --tags* werden alle Tags mit übertragen:

```
$ git push ../git-sample.git 1.00
Counting objects: 1, done.
Writing objects: 100% (1/1), 162 bytes | 0
bytes/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To ../git-sample.git
  * [new tag]    1.00 -> 1.00
```



Für die Anlage eines Accounts werden nicht viele Daten abgefragt (Bild 3)

Zum Start eines neuen Projekts ist noch alles ganz einfach, alle Entwickler arbeiten auf Version 1 hin. Sobald die Version aber ausgeliefert ist, wird ein Teil des Teams bereits an Version 2 arbeiten und ein anderer Teil wird noch Fehler in Version 1 beheben. Auch die Fehlerbehebungen und die Quellen von Version 2 müssen natürlich im gleichen Repository gehalten werden. Es gibt nun mehrere Möglichkeiten, mit solchen Verzweigungen umzugehen.

Oft sind Verzweigungen gewünscht, wenn zum Beispiel eine Entwicklergruppe schon neue Features umsetzt, während andere Entwickler die aktuell ausgelieferte Version pflegen. Damit man sich im Versionsgraphen auch zurechtfindet, werden Verzweigungen benannt. Die benannten Verzweigungen werden in Git als Branch bezeichnet.

In einem Git-Repository gibt es stets einen aktiven Branch. Bereits nach dem Anlegen eines Repositorys gibt es den Hauptentwicklungszweig mit dem Namen *master*. Die Branches eines Repositorys kann man sich über den Befehl *git branch* anzeigen lassen. Der aktive Branch ist dabei mit einem *-Zeichen gekennzeichnet. Um einen Branch zu wechseln, setzt man den Befehl *git checkout* in Verbindung mit dem Branchnamen ab. Möchte man auf Basis des aktuellen Commits einen Branch erzeugen, so muss man den Befehl *git branch branchname* absetzen. Zum Beispiel erzeugt der Befehl *git branch feature1* den Branch mit dem Namen *feature:*

```
$ git branch feature1
$ git branch
  feature1
* master
```

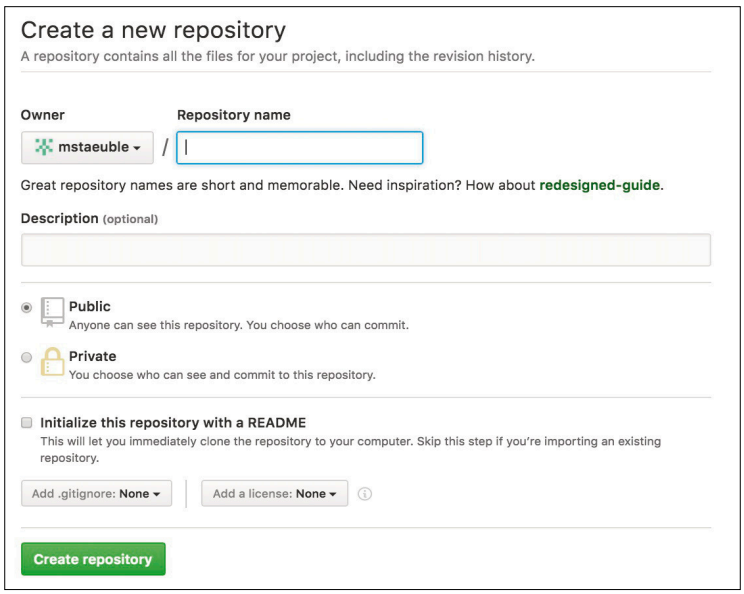
Möchte man nicht vom aktuellen Commit abzweigen, sondern von einem anderen Commit, so muss man den Hashwert des Commits angeben, zum Beispiel *git branch feature2 f5d-7f1f*. Auch von einem vorhandenen Branch kann man wiederum abzweigen, zum Beispiel wird mit *git branch feature3 feature2* ein neuer Branch *feature3* auf Basis des Branch *feature2* erzeugt. Um den Branch zu wechseln, muss man das Kommando *git checkout,* gefolgt vom Namen des Branchs, absetzen. Man kann auch direkt einen Branch erzeugen und zu diesem wechseln:

```
$ git checkout feature1
Switched to branch 'feature1'
$ git branch
* feature1
  master
```

Sobald man auf einem Branch entwickelt und einen weiteren Commit absetzt, so wandert auch der Branchzeiger mit. Das bedeutet, dass man beim Checkout den zuletzt auf dem Branch abgesetzten Commit erhält. Man kann einen Branch auch wieder löschen. Hierfür muss man das Kommando *git branch -d branchname* absetzen. Wenn mehrere Teams an unterschiedlichen Funktionalitäten in mehreren Branches arbeiten, müssen diese Branches auch irgendwann wieder in den HEAD integriert werden. Bild 2 zeigt zwei Branches: *master* und *feature*. Der Branch *feature* wurde von Master beim Commit C2 abgezweigt und es wurden zwei Commits C4 und C5 in diesem Branch getätigt. Auch im Branch *master* ging es mit Commit C3 weiter. Um nun den Branch *feature* in *master* zu integrieren, muss der Befehl *git merge feature* abgesetzt werden. Der Workspace muss dabei im Branch *master* sein. Git hat einen sehr intelligenten Algorithmus und versucht selbstständig, die Dateien zusammenzuführen. Sollte dies nicht gelingen, so zeigt Git die Konflikte an und diese müssen dann manuell gelöst werden.

Arbeiten mit GitHub

Nachdem Sie nun die wichtigsten Grundlagen von Git erlernt haben, geht es im Folgenden an die Nutzung von GitHub. Um GitHub selbst nutzen zu können, muss man sich zuerst einen GitHub-Account anlegen (Bild 3). ▶



Mit wenigen Angaben kann ein Online-Repository erzeugt werden (Bild 4)

Index umgehen

Man kann auch direkt mit dem lokalen Repository arbeiten und den Index überspringen. Um direkt Dateien aus dem Arbeitsverzeichnis ins lokale Repository zu schreiben, muss der Befehl

```
git commit -a
```

abgesetzt werden. Hier werden alle lokalen Änderungen direkt ins lokale Repository geschrieben. Mit

```
git commit text1.txt -a
```

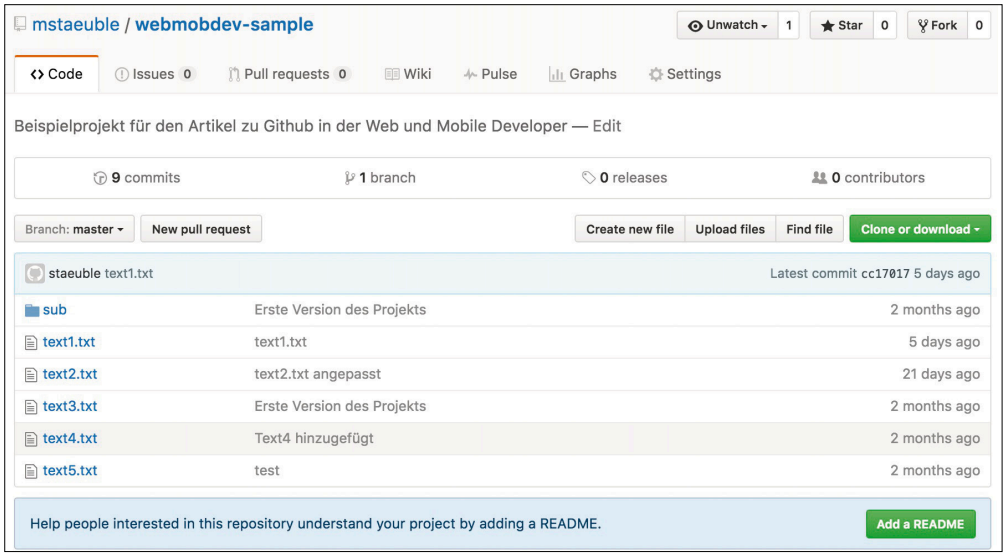
wird nur die Datei *text1.txt* ins lokale Repository geschrieben. Um den Inhalt von *HEAD* ins lokale Arbeitsverzeichnis zu kopieren muss der Befehlt *git checkout HEAD* abgesetzt werden.

Wenn Sie ein lokales Git-Projekt auf GitHub verwalten wollen, müssen Sie die beiden Projekte miteinander verbinden. Hierfür müssen Sie zunächst ein neues Repository in GitHub erzeugen. Klicken Sie hierfür auf der Startseite auf die Schaltfläche *Start a project*. Darüber können Sie ein neues Repository in Ihrem GitHub-Bereich anlegen. Hierfür müssen Sie einen Namen für das Repository, eine optionale Beschreibung und die Sichtbarkeit angeben (Bild 4). Zusätzlich können Sie für das Projekt direkt eine Lizenz auswählen und auch eine leere *README*-Datei erzeugen lassen. Die *README*-Datei wird beim Aufruf des Projekts im Browser angezeigt und soll eine kurze Zusammenfassung des Projekts darstellen.

Das Beispielprojekt trägt den Namen *webmobdev-sample* und ist über den URL https://github.com/mstaeuble/webmobdev-sample.git erreichbar. Nun soll dieses Repository mit einem lokalen Respository verbunden werden. Hierfür muss das GitHub-Repository dem lokalen System bekannt gemacht werden. Über *git remote add origin https://github.com/mstaeuble/webmobdev-sample.git* wird der URL dem System bekannt gemacht. Nun kann das lokale Repository über den Befehl *git push origin master* hochgeladen werden (Listing 3). Über den Browser kann dann das GitHub-Repository betrachtet werden (Bild 5).

An anderen GitHub-Projekten mitarbeiten

Um auch große Projekte mit vielen verteilten Entwicklern zu verwalten, gibt es in GitHub die sogenannten Pull-Requests. Dies sind Anfragen an den Verwalter des Repositorys. Der Verwalter kann die vorgeschlagenen Änderungen dann übernehmen. Um einen Pull-Request zu erzeugen, sind mehrere Schritte erforderlich. Zuerst muss man einen Fork eines vorhandenen Repositorys in seinen eigenen GitHub-Bereich erzeugen. Per Klick auf die Schaltfläche *Fork* kann man einen Fork erzeugen (Bild 6).



Auch die vergebenen Kommentare werden mit übertragen und angezeigt (Bild 5)

Änderungen per Zwischenrepository austauschen

Da man gewöhnlich nicht Zugriff auf ein lokales Repository eines Teammitglieds hat, müssen die Änderungen an eine zentral erreichbare Stelle ausgelagert werden. Dies sollte ein Repository sein, auf dem kein Entwickler arbeitet. Hier bietet sich an, ein sogenanntes Bare-Repository anzulegen. Dies ist ein Repository ohne Arbeitsverzeichnis. Ein solches Repository wird über den Parameter *--bare* beim Klonen angelegt, also mit *git clone --bare git-sample git-sample.git*. Änderungen eines Entwicklers werden über *git push* in das Bare-Repository hochgeladen. Ein anderer Entwickler holt sich die Änderungen dann über *git pull* wieder ab. Bei der Verwendung von *git push* muss neben dem Namen des Zielrepositorys noch der Name des Branchs angegeben werden. Der Name des Hauptentwicklungszweigs lautet *master*. Um diesen Zweig hochzuladen, müssen Sie den Befehl *git push ../git-sample.git master* eingeben.

Listing 3: Lokales Projekt in GitHub hochladen

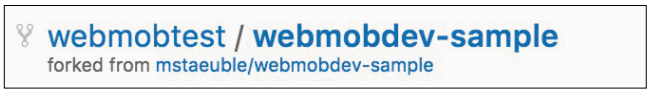
```
$ git status
On branch master
nothing to commit, working directory clean
$ git remote add origin https://github.com/
mstaeuble/webmobdev-sample.git
$ git push origin master
Counting objects: 26, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (19/19), done.
Writing objects: 100% (26/26), 1.91 KiB | 0 bytes/s,
done.
Total 26 (delta 8), reused 0 (delta 0)
remote: Resolving deltas: 100% (8/8), done.
To https://github.com/mstaeuble/webmobdev-sample.git
  * [new branch]      master -> master
```

Die Schaltfläche Fork befindet sich in der Kopfzeile zu einem Projekt (Bild 6)

Nach kurzer Zeit ist ein Fork von dem ausgewählten Projekt erzeugt. Neben dem Projektnamen erscheint das Fork-Symbol, und unter dem Namen wird die Quelle des Forks angegeben (Bild 7). Nun kann das Projekt über den Befehl *git clone* auf den lokalen Rechner heruntergeladen werden:

```
git clone https://github.com/webmobtest/
webmobdev-sample.git
```

Ein Pull-Request erfolgt immer in einem Branch. Deswegen wird zunächst ein neuer Branch unter dem Namen *bugfix* mit dem Befehl *git checkout -B bugfix* erzeugt. Dann wird die ▶



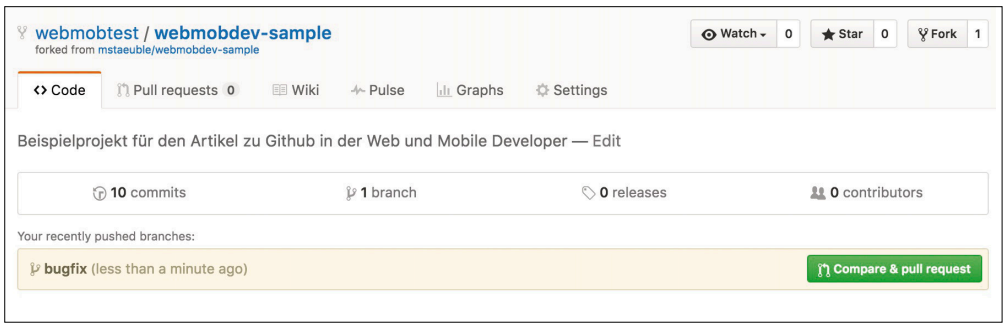
Forks erkennt man direkt in der Kopfzeile eines Projekts (Bild 7)

Zwischenspeicher Stash sichert Arbeitsergebnisse

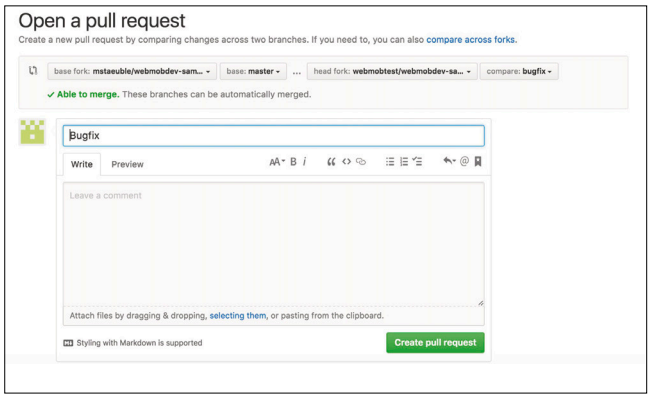
Als Entwickler im Projekt kommt man nicht darum herum, auch Wartungsarbeiten im Tagesbetrieb zu übernehmen. Da kann es sein, dass man kurzfristig einen Bugfix umsetzen muss und so die Arbeit unterbrechen muss. Wenn man die Änderungen aber noch nicht in das Repository schreiben möchte, so kann man diese in den Zwischenspeicher von Git (Stash) ablegen. Über *git stash --include-untracked* wird der aktuelle Arbeitsbereich (Workspace) in den Stash gelegt. Über *git stash pop* kann man sich die letzten Änderungen aus dem Stash wieder in den Arbeitsbereich laden. Möchte man sehen, was aktuell im Stash liegt, kann man sich dies über *git stash list* anzeigen lassen. Sollten mehrere Versionen im Stash liegen, so kann man sich ältere Versionen über einen Index wieder hervorholen, zum Beispiel über *git stash pop stash @{2}*. Um den Zwischenspeicher zu löschen, muss man den Befehl *git stash clear* absetzen.

Listing 4: Branch mit Änderungen hinzufügen

```
$ git checkout -B bugfix
Switched to a new branch 'bugfix'
$ touch test7.txt
$ vi test5.txt
$ git add *
$ git commit -m "test7.txt and test5.txt"
[bugfix 405efc6] test7.txt and test5.txt
  2 files changed, 1 insertion(+), 1 deletion(-)
  create mode 100644 test7.txt
$ git push
Everything up-to-date
$ git push origin bugfix
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 266 bytes | 0 bytes/s,
done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with
1 local objects.
To https://webmobtest@github.com/webmobtest/
webmobdev-sample.git
  * [new branch]      bugfix -> bugfix
```



Der Branch bugfix wird für einen Pull-Request angeboten (Bild 8)



Ein Pull-Request ist wie eine E-Mail mit Anhang, deshalb enthält der Dialog auch ein großes Beschreibungsfeld (Bild 9)

GitHub ohne Git

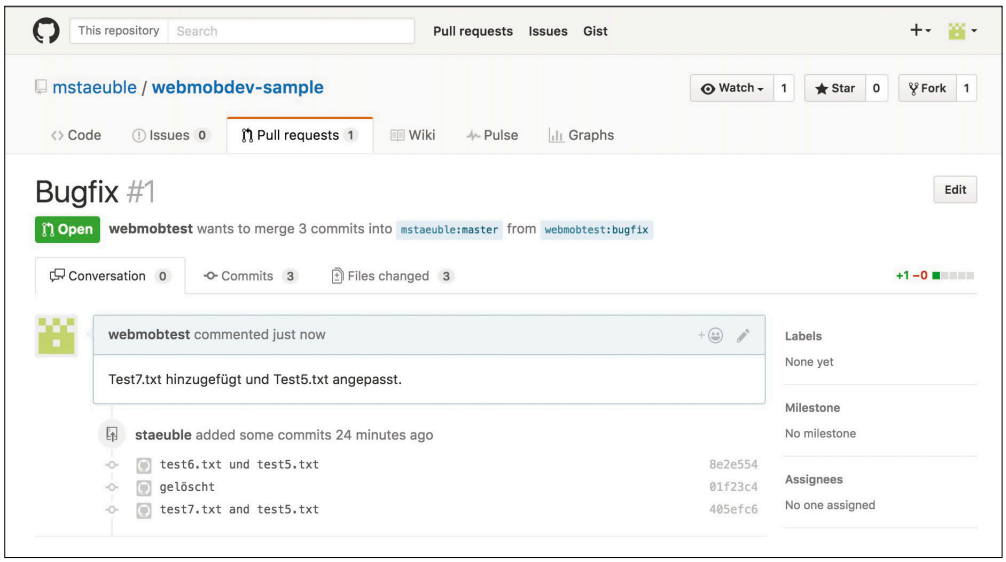
Nicht jeder hat auf seinem Rechner Git installiert und eingerichtet. Deswegen muss man nicht auf die Projekte in GitHub verzichten, denn auch ohne Nutzung von Git kann ein Projekt aus dem Online-Repository heruntergeladen werden. Nachdem das gewünschte Projekt in GitHub im Browser geöffnet ist (per Klick auf den Namen des Projekts). erscheint in der Menüzeile die grüne Schaltfläche *Clone or download*. Bei Klick auf die Schaltfläche erscheint die Option *Download Zip*. Darüber kann das Projekt als ZIP-Archiv heruntergeladen werden. Möchte man einen bestimmten Branch herunterladen, so muss man diesen Branch zuvor über das Menü *Branch* auswählen.

Datei *test7.txt* angelegt sowie die Datei *test5.txt* angepasst und wieder zum lokalen Repository hinzugefügt. Anschließend werden die Änderungen direkt an GitHub übertragen (Listing 4).

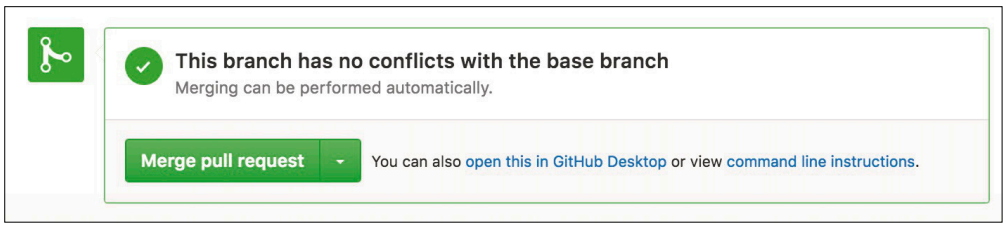
In der Projektübersicht erscheint nun die grüne Schaltfläche *Compare & pull request* (Bild 8). Nach einem Klick auf die Schaltfläche kann der Pull-Request noch näher beschrieben werden (Bild 9). Nach dem Klick auf die Schaltfläche *Create Pull request* wird dieser in einer Zusammenfassung dargestellt (Bild 10). Im Originalprojekt wird nun die Schaltfläche *Merge pull request* dargestellt. Darüber können die Änderungen ins Projekt übernommen werden (Bild 11).

Dokumentation mit GitHub erstellen

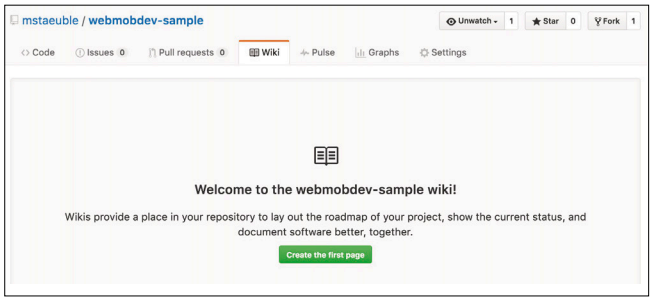
Neben den Projektdateien gehört auch eine Dokumentation zu einem Projekt. Mindestens die erwähnte *README*-Datei sollte in einem GitHub-Projekt liegen. GitHub stellt aber auch die Möglichkeit zur Pflege eines Wikis zur Verfügung. Direkt unter dem Projektnamen gibt es eine Liste mit Reitern. Per Klick auf die Schaltfläche *Create the first page* kann die



Der erstellte Pull-Request erscheint nochmals in einer Zusammenfassung (Bild 10)

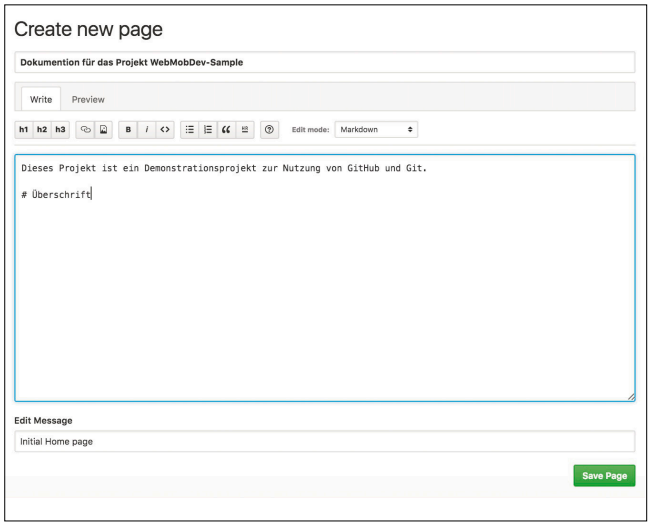


Der Verwalter des Original-Repositorys erhält eine E-Mail bei einem neuen Pull-Request und kann über eine Schaltfläche die Änderungen übernehmen (Bild 11)

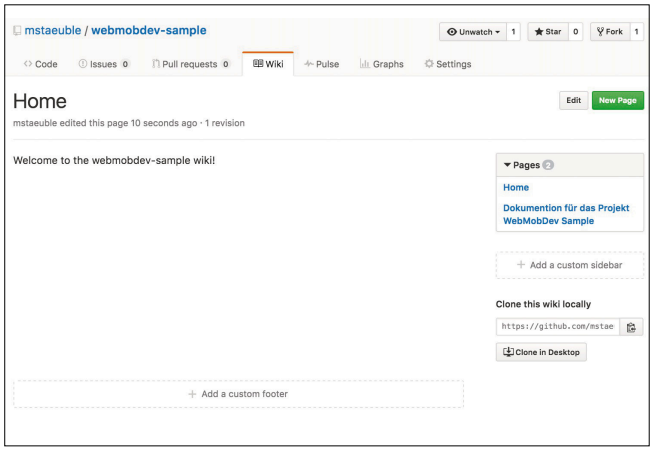


Für den Start eines Wikis genügt der Klick auf die grüne Schaltfläche (Bild 12)

Ein GitHub-Wiki ist auch nur ein Git-Projekt (Bild 15)



Der Editor von GitHub ähnelt dem Editor anderer Wikis (Bild 13)



Bisher enthält das Wiki nur die Hauptseite und die gerade erstellte Seite, die über das Seitenmenü geöffnet werden kann (Bild 14)

erste Seite eines Projektwikis erzeugt werden (Bild 12). Nun öffnet sich der Editor für die Erstellung der Seite. Über ein Menü können die einzelnen Formate (zum Beispiel *Überschrift h1)* ausgewählt werden. Der Editor stellt den Text direkt in der Beschreibungssprache dar (Bild 13).

Der Editor unterstützt mehrere Beschreibungssprachen; diese können über das Feld *Edit mode* ausgewählt werden. Folgende Beschreibungssprachen werden unterstützt: AsciiDoc, Creole, Markdown, MediaWiki, Org-mode, Pod, Rdoc, Textile und reStructuredText. Der Editor hat neben der Quellcode-Ansicht auch eine Vorschau (Reiter *Preview)* zur Ansicht der endgültigen Seite.

Nach dem Speichern der Seite gelangt man zur Hauptseite des Projektwikis. Diese Hauptseite kann man auch anpassen. Daneben kann eine Fußzeile und eine Seitenleiste erstellt werden (Bild 14).

Ein Projektwiki in GitHub ist ebenfalls ein Git-Projekt und kann damit auch auf den lokalen Rechner geladen werden. Der URL endet dabei auf *.wiki.git* anstatt nur auf *.git*. Für das Beispielprojekt lautet der URL https://github.com/mstaeuble/webmobdev-sample.wiki.git. Nach dem Herunterladen des Projekts über *git clone https://github.com/mstaeuble/webmobdev-sample.wiki.git* liegen die zwei Wikiseiten im Verzeichnis *webmobdev-sample.wiki* (Bild 15).

An dem Beispielprojekt sieht man auch, dass der Titel der Wikiseite direkt zum Dateinamen wird, Leerzeichen werden dabei durch einen Bindestrich dargestellt. Wie schon erwähnt, ist das Wiki nichts anderes als ein Git-Projekt, daher las- ▶

Zweiter Account für Tests erforderlich

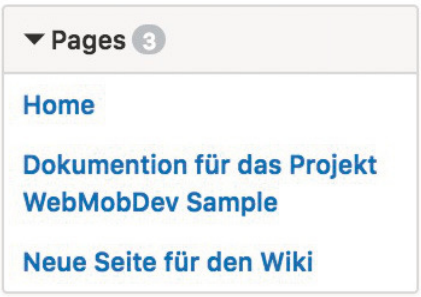
Möchten Sie das Forken selbst auf Ihrem eigenen Projekt testen, so benötigen Sie einen zweiten Account und ein zweite E-Mail-Adresse. Denn es ist nicht möglich, einen Fork von Ihrem eigenen Projekt im selben GitHub-Account zu haben. Für diesen Test wurde der Benutzer *webmobtest* erstellt. Sein GitHub ist erreichbar unter https://github.com/webmobtest.

Listing 5: Wiki kann direkt lokal mit Git-Kommandos aktualisiert werden

```
$ touch Neue-Seite-für-das-Wiki.md
$ git add *
$ git commit -m "Neue Seite"
[master 70d94b9] Neue Seite
  2 files changed, 1 insertion(+), 1 deletion(-)
  create mode 100644
"Neue-Seite-f\303\274r-das-Wiki.md"
Markuss-MBP-2:webmobdev-sample.wiki mstaeuble$ git
push
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 396 bytes | 0 bytes/s,
done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/mstaeuble/webmobdev-sample.wiki.
git
  d2ff028..70d94b9  master -> master
```

Links zum Thema

- AsciiDoc
  www.methods.co.nz/asciidoc
- Atlassian
  https://de.atlassian.com
- Creole
  www.wikicreole.org
- CVS
  http://savannah.nongnu.org/projects/cvs
- Git
  https://git-scm.com
- GitHub
  https://github.com
- GitHub Markup
  https://github.com/github/markup
- GitHub Wikis
  https://help.github.com/articles/about-github-wikis
- Markdown
  http://markdown.de
- MediaWiki
  https://www.mediawiki.org/wiki/MediaWiki/de
- Org-mode
  http://orgmode.org
- Pod
  https://metacpan.org/pod/Pod::Simple::Wiki
- Rdoc
  http://docs.seattlerb.org/rdoc
- reStructuredText
  http://docutils.sourceforge.net/rst.html
- SourceTree
  https://www.sourcetreeapp.com
- Subversion
  https://subversion.apache.org
- Textile
  https://txstyle.org



Die Seite Neue Seite für das Wiki ist nun auch in der Seitenleiste zur Auswahl aufgeführt (Bild 16)

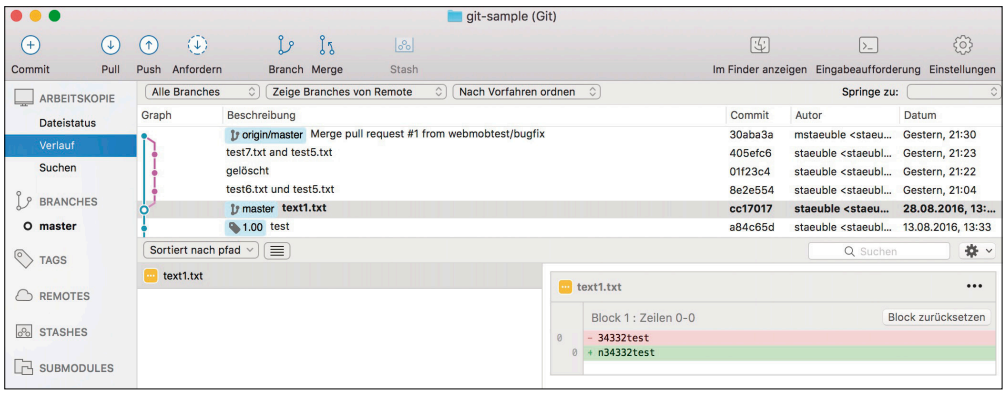
Oberfläche von SourceTree nach dem ersten Start (Bild 17)

Projekt git-sample in der Übersicht von SourceTree (Bild 18)

sen sich auch lokal Änderungen an den Seiten vornehmen und neue Seiten erzeugen (Listing 5). Solche Änderungen sind direkt im Browser sichtbar. Die neu erstellte Seite erscheint dabei im Menü der Seitenleiste (Bild 16).

Kommandozeile verstehen

Git lässt sich gut über die Kommandozeile bedienen, mehr Komfort bieten aber grafische Git-Clients. Ein sehr guter Client ist SourceTree von Atlassian. SourceTree steht kostenlos für die Betriebssysteme Windows und Mac OS X zur Verfügung. Die Installation verläuft problemlos. Nach dem Start



Das Arbeitsfenster von SourceTree bietet großen Komfort in der Arbeit mit Git (Bild 19)

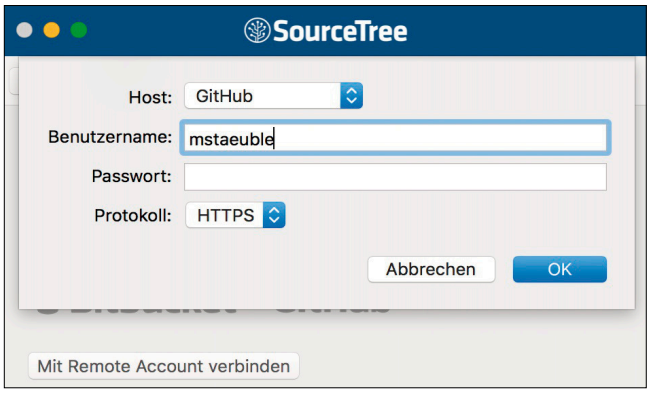
stellt sich SourceTree noch spartanisch dar (Bild 17). Zuerst muss man ein Repository über die Schaltfläche + *Neues Repository* hinzufügen. Hier ist zu unterscheiden, ob es sich um ein lokales Repository handelt oder um ein Remote-Repository. SourceTree bietet die Möglichkeit, direkt ein Verzeichnis per Drag and Drop hinzufügen. Ist das Verzeichnis ein Git-Projekt, so wird es direkt zu SourceTree hinzugefügt (Bild 18).

Nach Doppelklick auf das Projekt öffnet sich das eigentliche Arbeitsfenster von SourceTree (Bild 19). Hier wird das Repository sehr anschaulich dargestellt, und über die Menüzeile stehen alle notwendigen Kommandos zur Verfügung.

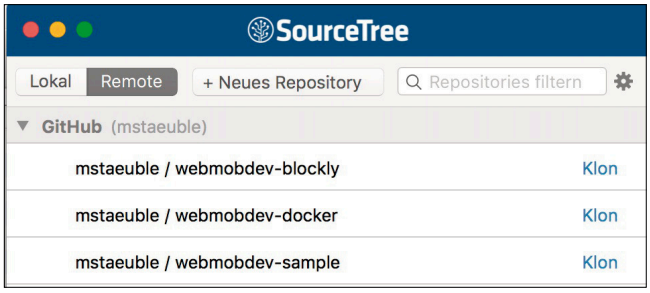
Wie eingangs erwähnt, lässt sich auch ein Remote-Repository mit SourceTree verbinden. Hierbei werden BitBucket und GitHub unterstützt (Bild 20).

Nach Eingabe der Accountdaten werden alle GitHub-Repositories des Accounts anzeigt (Bild 21). Über einen Klick auf ein Projekt öffnet sich direkt der Dialog, um das Projekt auf das lokale System zu klonen. Hier muss der Zielpfad angegeben werden, dann wird das Projekt geklont und erscheint unter den lokalen Repositories. Per Klick auf das Repository öffnet sich dieses im Arbeitsfenster von SourceTree.

Mit Git und GitHub gibt es eigentlich keine Ausrede mehr, warum man sein Projekt nicht versioniert. Denn wem ist es nicht schon einal passiert, das man Änderungen vom Vortag rückgängig machen möchte oder eine bestimmte Version eines Dokuments benötigt. Mit Git ist es sehr einfach, zu einem bestimmten Stand zurückzuspringen. Wem GitHub zu öffentlich ist, der kann auch nur mit lokalen Repositories arbeiten und diese in ein Backup einbeziehen. Git und GitHub beziehen sich nicht nur auf Entwicklungsprojekte, sondern können auch zur Verwaltung anderer Dateien, zum Beispiel Dokumentationen, genutzt werden. Schon Git auf dem lokalen Rechner ist ein klarer Sprung in der Produktivität. ■



Benutzername und Passwort sind für den Zugriff auf GitHub erforderlich (Bild 20)



Alle GitHub-Projekte eines Accounts im Überblick (Bild 21)

In ein fremdes Repository schreiben

Bei der Nutzung von *git push* wird automatisch der konfigurierte Benutzername verwendet. Möchte man einen Push unter einen anderem Namen ausführen, so muss man dies konfigurieren. Über *git config remote.origin.url https://webmobtest@github.com/webmobtest/webmobdev-sample.git* wird nun für den URL https:// github.com/webmobtest/webmobdev-sample.git der Benutzername *webmobtest* genutzt. Beim Aufruf von *git push* wird man nun nach dem Passwort gefragt, und nach dessen richtiger Eingabe wird der Push durchgeführt.

Dr. Markus Stäuble
ist Informatiker, Conference Chair der Developer Week und Programmleiter Make beim Franzis Verlag. Neben Make beschäftigt er sich viel mit dem Thema Mobile und hat zu dessen Auswirkungen auf die Arbeitswelt promoviert.
https://github.com/mstaeuble

CROSS-PLATFORM DEVELOPMENT MIT RAPIDCLIPSE

Visuelle Java-Entwicklungsumgebung

RapidClipse ist eine neue Eclipse-Distribution, die aus Eclipse eine visuelle Java-Entwicklungsumgebung macht und Cross-Platform Development ermöglicht.

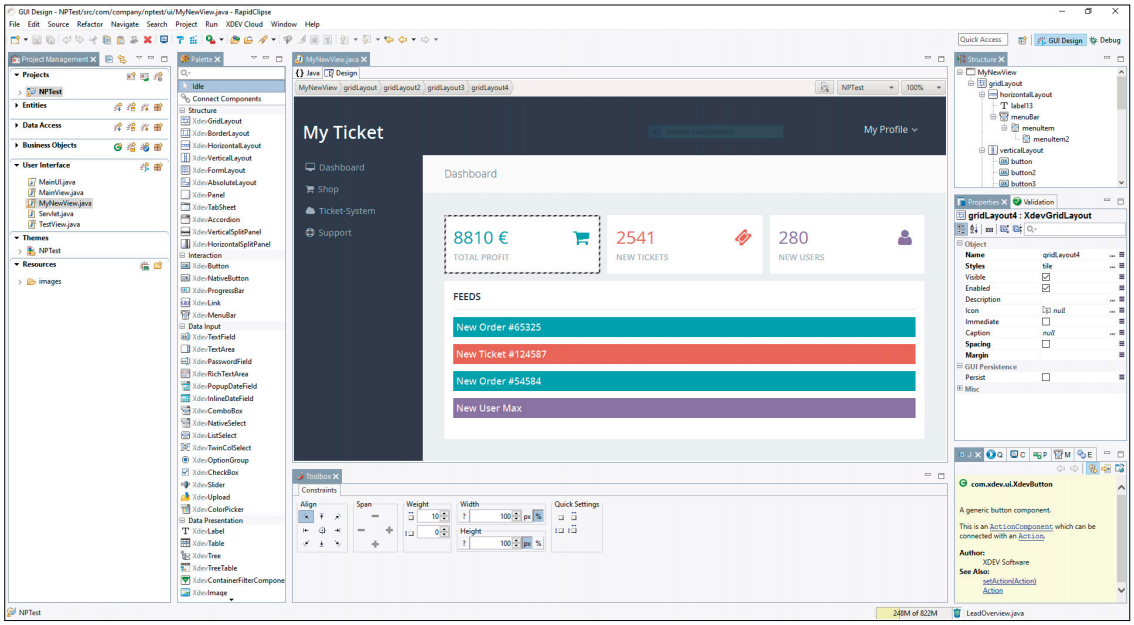
Jede Anwendung lässt sich mit RapidClipse als HTML5-Webanwendung, mobile App, klassische Java-Desktop-Applikation oder Cloud-Anwendung deployen. Der Funktionsumfang ist beeindruckend und die Entwicklung verblüffend einfach.

Java und Eclipse ist eine Kombination, auf die Java-Profis schwören und die praktisch den Industriestandard darstellt. Trotzdem machen viele Anwendungsentwickler, insbesondere Webentwickler, einen weiten Bogen darum und können weder mit Java noch mit Eclipse etwas anfangen. Java ist viel zu kompliziert und Eclipse aufgrund der fehlenden visuellen Unterstützung für Java-Ein- und -Umsteiger schlicht und einfach viel zu komplex. Viele an Java grundsätzlich stark interessierte Anwender geben schon nach kurzer Einarbeitungszeit frustriert wieder auf (Bild 1).

Mit RapidClipse ändert sich das jetzt schlagartig. RapidClipse ist eine völlig neue Eclipse-Distribution, die Eclipse zu einer visuellen und komfortablen Java-Entwicklungsumgebung macht. Der Oberflächendesigner erinnert an Visual Basic und Delphi und ermöglicht die Erstellung responsiver HTML5-Oberflächen, ohne dass man dafür eine Zeile Code schreiben muss. RapidClipse geht jedoch noch sehr viel weiter, denn es vereinfacht auch die Anbindung beliebiger Datenbanken erheblich und beschleunigt dadurch die Entwicklung von Datenbankapplikationen um ein Vielfaches.

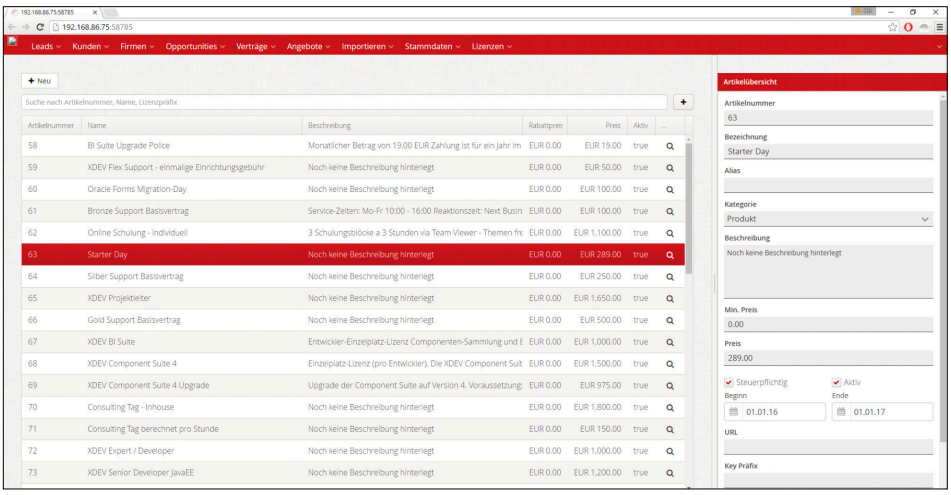
Zudem unterstützt RapidClipse Cross-Platform Development. Jedes Projekt lässt sich aus ein und derselben Codebasis heraus als HTML5-Webanwendung, mobile App oder klassische Java-Desktop-Applikation deployen. Mit dem integrierten Cloud-Deployment kann man Anwendungen per Mausklick in die Cloud deployen und in Sekundenschnelle über das Web verfügbar machen.

RapidClipse setzt auf Standards und im Java-Umfeld anerkannte Tools und Frameworks auf. Für die Oberflächenentwicklung findet Vaadin Verwendung, das wiederum auf dem Google Web Toolkit aufsetzt. Für die Datenbankentwicklung kommen JPA und Hibernate mit den JBoss-Tools zum Einsatz. Mit RapidClipse entwickelte Anwendungen basieren auf der Java Standard Edition (Java SE). Das hat zum einen den Vorteil, dass der Entwickler kein Java-Enterprise-Know-how (Java EE) besitzen muss und zum anderen als Laufzeitumgebung kein Java-EE-Application-Server benötigt wird.



RapidClipse ist eine Eclipse-Distribution mit HTML5-GUI-Builder, verbesserten Hibernate-Tools sowie Cross-Platform- und Cloud-Deployment-Tools (Bild 1)

Ziel des RapidClipse-Projekts ist es, die gesamte Anwendungsentwicklung mit Java und Eclipse radikal zu vereinfachen und zu beschleunigen. Zielgruppe sind Anwendungsentwickler aus dem Web- und 4GL-Umfeld, die schnell und einfach auf Java umsteigen, moderne Business-Anwendungen auf Basis von Java entwickeln oder wichtige Altanwendungen schnell und kostengünstig auf Java portieren möchten (Bild 2).



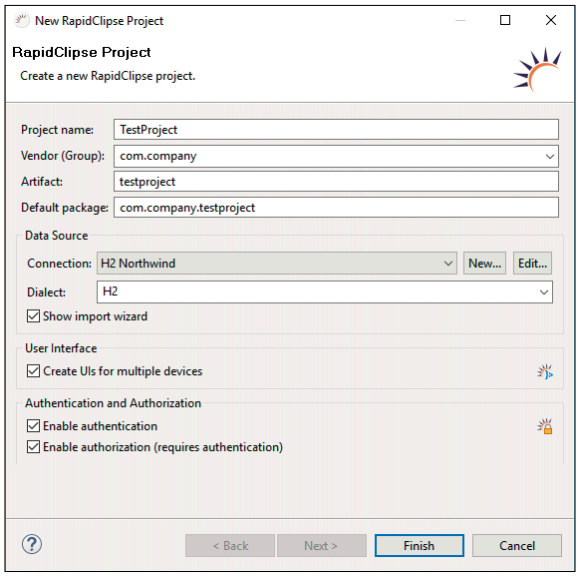
Mit RapidClipse lassen sich moderne Business-Anwendungen auf Basis von Java-Standards in Rekordzeit entwickeln (Bild 2)

Einfache Installation

Eclipse ist eine Entwicklungsplattform für zahlreiche Programmier- und Skriptsprachen, Frameworks und andere Tools. Das Angebot an Plug-ins ist enorm. Wer eine Webanwendung oder App entwickeln möchte, muss zuerst einmal die dazu erforderlichen Plug-ins zusammensuchen, installieren und per XML-Konfiguration so aufeinander abstimmen, dass diese sich nicht gegenseitig in die Quere kommen. So gut wie jeder Einsteiger hat damit erhebliche Probleme.

Mit RapidClipse fällt diese Einstiegshürde vollständig weg. Als Eclipse-Distribution liefert RapidClipse alle wichtigen Plug-ins, die man für die Anwendungsentwicklung mit Java braucht, bereits im vorkonfiguriertem Zustand mit, unter anderem die Java EE Tools, die JBoss Hibernate Tools, Vaadin, die Versionsverwaltung Git sowie die zusätzlichen RapidClipse Eclipse IDE Tools. Nach der Installation kann man sofort starten.

Für einen einfachen Projektstart sorgt der Projektassistent, der alle benötigten Java-Bibliotheken per Maven von einem zentralen Repository lädt und automatisch sämtliche XML-Konfigurationen setzt, sodass man sich damit nicht auseinandersetzen muss (Bild 3).



Der Projekt-Assistent lädt via Maven automatisch die aktuelle Version aller benötigten Java-Bibliotheken inklusive Abhängigkeiten (Bild 3)

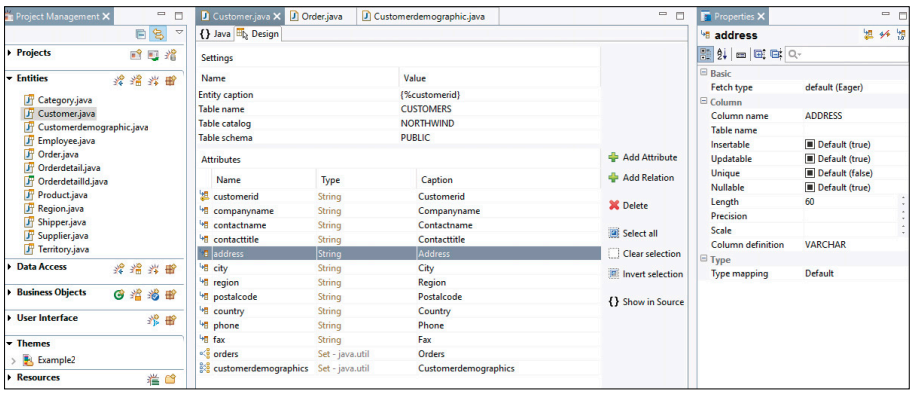
Java Persistence API

Datenbankzugriffe werden in RapidClipse nach dem JPA-Standard (Java Persistence API) realisiert. Danach werden Abfragen nicht direkt in SQL, sondern in Java formuliert, und der Entwickler kann objektorientiert auf relationale Daten zugreifen. Erst zur Laufzeit werden SQL-Statements von einem sogenannten ORM-Framework (Objekt-Relationales Mapping) erzeugt und an den Datenbankserver gesendet. Das bringt im Vergleich zu Plain-SQL-Strings enorme Vorteile mit sich, unter anderem eine praktische Eingabevervollständigung im Code-Editor, Hinweise auf Fehler in der Abfrage in Form von Compiler-Warnungen, die Möglichkeit, Abfragen debuggen und automatisierte Softwaretests mit DB-Unit durchführen zu können, sowie vollständige Datenbankunabhängigkeit. Als ORM-Framework setzt RapidClipse auf Hibernate, das als De-facto-Standard gilt. Die Verwendung von SQL-Strings und Stored Procedures ist jedoch auch mit Hibernate grundsätzlich möglich.

Da die Hibernate-Programmierung als enorm komplex gilt und Java Einsteiger damit völlig überfordert wären, wurde für RapidClipse die gesamte Datenbankentwicklung mit Hibernate drastisch vereinfacht. Auf Seiten der IDE wurden die JBoss Hibernate Tools gleich an mehreren Stellen optimiert und mit zusätzlichen Tools, unter anderem um einen Entity-Editor, mit dem man sein gesamtes Entity-Modell sehr schnell erstellen kann, erweitert. Besonders gefallen hat uns in diesem Zusammenhang die Definition von Relationen.

Nicht nur Foreign-Key-Constraints, sondern sogar die Kardinalität (1:1, 1:n, n:1, n:m), bei der sich selbst erfahrene Entwickler manchmal nicht ganz sicher sind, wird automatisch korrekt gesetzt, sodass man beim Datenbankdesign praktisch keine folgenschweren Fehler machen kann. Sobald das Datenmodell steht, kann man sich per Export-Funktion abschließend die gesamte Datenbank vollautomatisch generieren lassen, was zugleich der für Hibernate empfohlene Standardweg ist (Bild 4). ▶

Möchte man dagegen auf eine bereits vorhandene Datenbank aufsetzen, kann man sich umgekehrt das gesamte Entity-Modell per Datenbank-Import-Funktion automatisch generieren lassen. Mit den JBoss Tools kommt es dabei jedoch fast immer zu Fehlern, da der Hibernate-Importer vor allem proprietäre Datenbank-Datentypen nicht korrekt auf entsprechende Java-Typen mappen kann. Hibernate-Nutzer müssen somit häufig zahlreiche Code-Anpassungen vornehmen und die korrekten Annotations selbst setzen. Wer jedoch kein Hibernate-Know-how besitzt und nicht ganz genau weiß, was er zu tun hat, kommt an dieser Stelle nicht mehr weiter. Mit RapidClipse funktioniert der erweiterte Hibernate-Import dagegen reibungslos, was eine enorme Vereinfachung ist und uns deshalb sehr begeistert hat (Bild 5).



Der Entity-Editor ermöglicht schnelles Datenbankdesign (Bild 4)

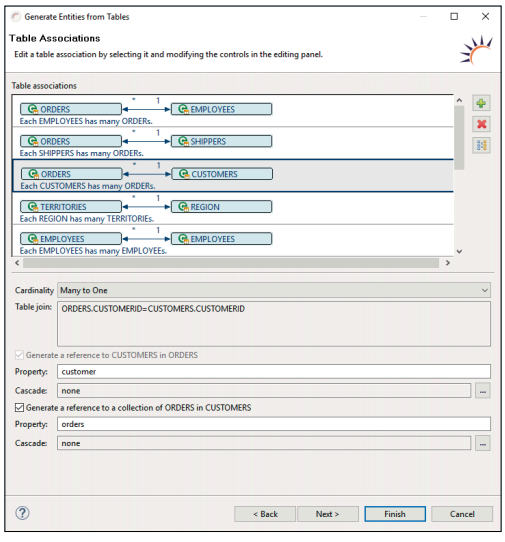
Durch den Einsatz von Hibernate werden von RapidClipse alle wichtigen Datenbanken unterstützt, unter anderem Oracle, DB2, MS SQL, MySQL, PostgreSQL et cetera. Für Testzwecke steht die Northwind-Testdatenbank auf Basis von H2, einer frei verfügbaren Java-Datenbank-Engine, zur Verfügung. Für das initiale Anlegen einer Datenquelle gibt es einen Assistenten.

Moderne HTML5-Oberflächen

Das Prunkstück von RapidClipse ist der GUI-Builder, der wie ein Grafikprogramm funktioniert. Beim Anlegen neuer Views kann man festlegen, ob man eine Oberfläche für den Desktop, ein Tablet oder Smartphone designen möchte. Alle in der Palette verfügbaren UI-Komponenten lassen sich dann per Drag and Drop in die Arbeitsfläche einfügen und mit Hilfe verschiedener Layouts anordnen und ausrichten.

Die Widget-Palette selbst ist sehr umfangreich. Neben den gängigen Formular-Controls stehen eine Table-Komponente, Tree, Tree-Table, Richtext-Editor, Datepicker, Accordion sowie Container für das Einbinden von Audio, Video, Flash und anderer Webinhalte zur Verfügung. Controls, die man miteinander verknüpft, können vollautomatisch miteinander interagieren. Häufig benötigte Master-Detail-Ansichten sind auf diese Weise mit wenigen Mausklicks umsetzbar (Bild 6).

Auch voll funktionsfähige Formulare kann man sich per Assistent generieren lassen (Bild 7). Neu ist die leistungsfähige Such- und Filter-Komponente mit Volltextsuche, mit der Endnutzer eigenhändig Filterbedingungen anlegen können.



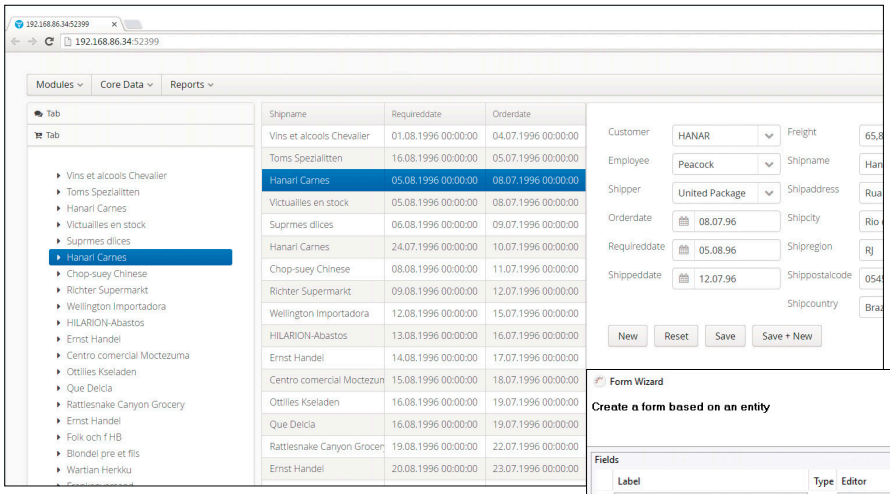
Beim stark verbesserten Hibernate-Import werden alle datenbankspezifischen Datentypen korrekt auf Java-Typen gemappt (Bild 5)

Die meisten anderen GUI-Builder unterstützen den Entwickler zwar bei der Erstellung von Oberflächen – darum, wie später Daten auf der Oberfläche angezeigt, editiert und persistiert werden, muss sich der Entwickler jedoch selbst kümmern. Nur ist gerade dieser Teil in Java am kompliziertesten, weshalb viele Java-Ein- und -Umsteiger genau an dieser Stelle nicht mehr weiterkommen. Wer noch keine Erfahrung mit JDBC, JPA, Hibernate, EJB, Spring oder Ähnlichem hat, wird unter normalen Umständen keine Datenbankanwendung in Java entwickeln können. Genau hier setzt das RapidClipse-Framework an, das dem Entwickler ein Data Binding zwischen Vaadin und Hibernate zur Verfügung stellt.

Damit werden aus nackten Oberflächen echte Datenbankanwendungen. Die Vorgehensweise ist denkbar einfach. Jedes Entity lässt sich über den GUI-Builder per Drag and Drop mit einer Table, List- oder Combobox verknüpfen. Um Datenbankzugriffe und Visualisierung, automatisiertes Nachladen (Lazy Loading), Persistieren editierter Daten et cetera kümmert sich das RapidClipse-Framework vollautomatisch.

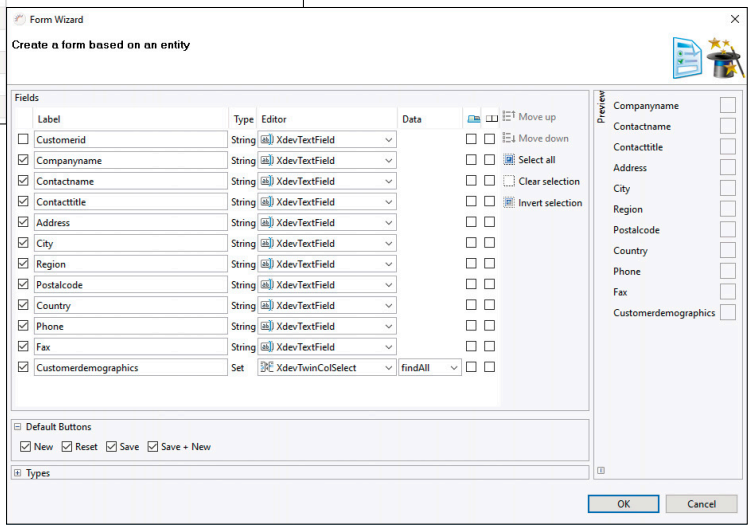
Die bedeutendste Vereinfachung bringt jedoch das Fully-Managed-Hibernate-Session-Konzept, durch das sich der Entwickler weder mit Hibernate-Sessions noch mit dem Entity-Lifecycle auseinandersetzen muss. Darüber hinaus enthält das Framework sämtliche Features, die in keiner vernünftigen Business-Anwendung fehlen dürfen, unter anderem ein Konzept für Authentifizierung und Autorisierung, Lösungen für konkurrierende Schreibzugriffe, eine Druckfunktion für Tabellen und vieles mehr.

Alles in allem bietet RapidClipse damit out of the Box eine vollständige Anwendungsinfrastruktur auf Basis einer modernen 3-Schichten-Architektur, die man als Grundlage für eine leistungsfähige und skalierbare Business-Applikation braucht. RapidClipse Entwickler sparen damit wertvolle Entwicklungszeit, die



Oberflächen mit Trees, Tabellen, Such- und Filterfunktion, Master-Detail-Views und Formularen erstellt man sekundenschnell per Drag and Drop und völlig ohne Code (Bild 6)

Per Formular-Assistent kann man sich für jedes Entity den passenden Stammdaten-Editor dazugenerieren lassen (Bild 7)



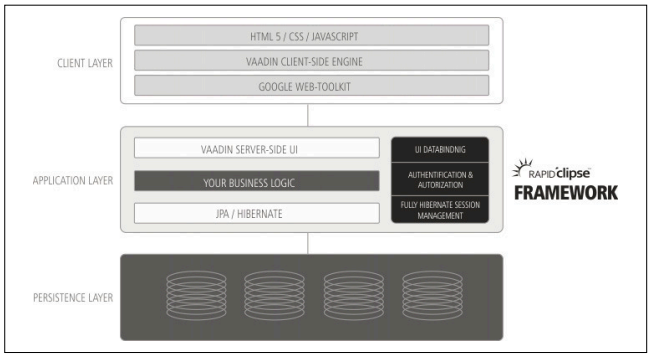
ansonsten für Planung, Implementierung, Tests und Wartung für eine Eigenentwicklung aufgebracht werden müsste (Bild 8).

Low-Level-Prozesse im Hintergrund

Bis die fertige HTML5-Oberfläche im Browser erscheint, muss das System im Hintergrund einiges an Arbeit leisten. Mit RapidClipse soll sich der Anwendungsentwickler zu 100 Prozent auf die Entwicklung von Anwendungs-Features konzentrieren können und sich ganz bewusst nicht mit den im Hintergrund ablaufenden Low-Level-Prozessen herumplagen müssen.

Bei jeder Änderung im GUI-Builder speichert RapidClipse die GUI-Beschreibung als XML-Metacode. Das funktioniert auch bidirektional. Wer sich mit Drag and Drop nicht anfreunden kann, sondern lieber deklarativ entwickelt, kann seine Oberflächen direkt in XML schreiben. Nach dem Speichern werden die Änderungen auch im GUI-Builder sichtbar. Aus dem XML-Code erzeugt RapidClipse sofort Java-Code auf Basis von Vaadin, der im Code-Editor angezeigt, jedoch nicht editiert werden kann.

RapidClipse-Anwendungen sind also zu 100 Prozent Java-Anwendungen, die auf dem Server ausgeführt werden (Servlet). Erst zur Laufzeit erzeugt das Vaadin Servlet dynamischen HTML5-, CSS- und JavaScript-Code und sendet diesen an den Client, der die Oberfläche im Browser rendert. Auch der permanente Datenaustausch zwischen Client und Server, DOM-Manipulationen und die Browser-History werden vollautomatisch von Vaadin gemanagt, was die Webentwicklung extrem vereinfacht. Die im Browser sichtbaren UI-Komponenten basieren auf dem Google Web Toolkit und können mit Hilfe von CSS individuell gestylt werden.

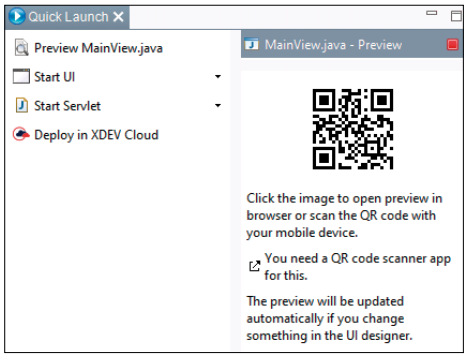


RapidClipse-Anwendungen basieren auf einer 3-Schichten-Architektur. Das Framework bietet ein Data Binding für die elegante Verbindung von GUI und Datenbank via Hibernate (Bild 8)

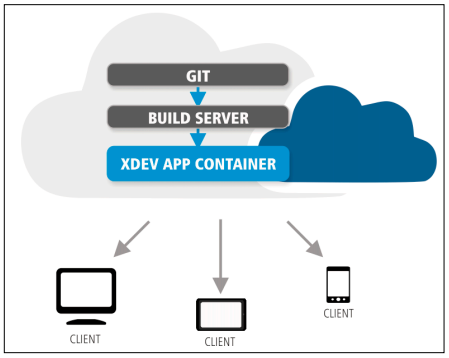
Einfaches Testen und Deployen

Mit RapidClipse ist man in der Lage, die eigene Anwendung per Mausklick auf dem lokalen Rechner zu testen. Für die Vorschau wird ein Jetty Server gestartet. Durch Scannen eines QR-Codes kann man dieselbe Vorschau sogar direkt auf einem Smartphone oder Tablet ausführen. Jede Änderung im GUI-Builder wird dabei binnen Sekunden auf das Gerät gepusht. Die fertige Anwendung lässt sich abschließend mit dem Deploy-Assistenten als HTML5-Webanwendung oder klassische Java-Desktop-Applikation fertigstellen (Bild 9).

Mit Version 3.0 steht das bislang bedeutendste Release bereits kurz vor der Freigabe. Ein erster Blick auf die Developer-Preview ist sehr vielversprechend. Als erstes Highlight der neuen Version fällt sofort der komplett neue GUI-Builder auf, der die bislang erst zur Laufzeit im Browser sichtbare HTML5-Oberfläche jetzt bereits im Entwurf anzeigt und damit zu 100 Prozent Wysiwyg umsetzt. Auch das Ändern einzelner Styles oder des gesamten Themes ist bereits im Entwurf sichtbar. ▶



Per Mausklick kann man einzelne Fenster oder die gesamte Anwendung lokal starten, als Webanwendung, Java-Desktop- oder mobile App deployen und in der Cloud ausführen (Bild 9)

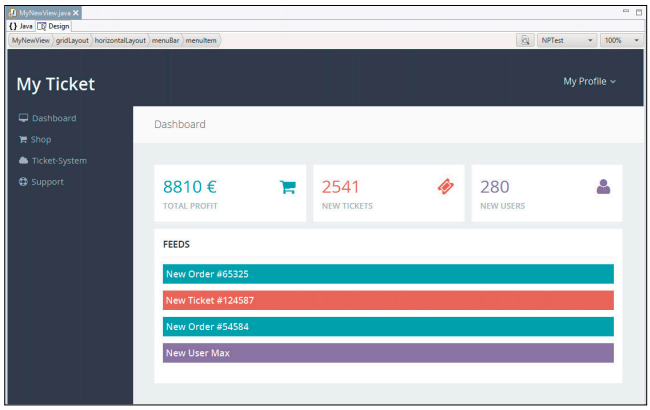


Mit der integrierten XDEV Cloud kann man Java-Anwendungen per Mausklick in der Cloud deployen, testen und produktiv hosten (Bild 11)

Besonders erfreulich ist, dass man im GUI-Builder jetzt genauso wie im Browser zoomen kann. Vor allem für das Designen kleinerer Seitenelemente ist das eine enorme Erleichterung. Die neuen GUI-Templates helfen vor allem dem Einsteiger, der noch keine genaue Vorstellung davon hat, wie seine Weboberflächen oder Mobile-Apps aussehen sollen. Menüs werden jetzt nicht mehr direkt im Entwurf konstruiert, sondern mit Hilfe eines Assistenten. Einige nervige Fehler, wie das Kollabieren von Layouts, treten mit dem neuen GUI-Builder nicht mehr auf. Bereits vorhandene Projekte lassen sich mit dem neuen GUI-Builder ohne Probleme öffnen und weiterbearbeiten (Bild 10).

SQL-Editor erzeugt Hibernate-Code

Ein weiteres Highlight ist der neue Query-Editor, der jeden Hibernate-Nutzer brennend interessieren sollte. Datenbankabfragen in Java zu formulieren hat, wie bereits erwähnt, jede Menge Vorteile im Vergleich zu Plain-SQL-Strings. Der große Nachteil ist jedoch die im Vergleich zu SQL deutlich höhere Komplexität. Zudem spricht für SQL, dass nahezu jeder Software- und Datenbankentwickler damit vertraut ist. Beide Vorteile sollen nun mit einem neuen SQL-Code-Editor für Eclipse kombiniert werden. Der Editor erlaubt dem Entwickler, sämtliche benutzerdefinierten Datenbankabfragen in Plain-SQL zu formulieren. Sogar Autovervollständigung und Syntax-Checks werden unterstützt. Im Hintergrund wird der SQL-Code dann in Hibernate-5-konformen Java-Code auf Basis des JPA Criteria API generiert.



Der neue GUI-Builder kann die HTML5-Oberfläche schon im Entwurf anzeigen (Bild 10)

Für das Starten der Vorschau und des finalen Deployments gibt es jetzt ein eigenes Fenster, das alle möglichen Varianten an einer zentralen Stelle zusammenfasst. Dies ist sehr praktisch, weil es bei einer Anwendung unterschiedliche Einstiegspunkte geben kann. Mit nur einem Mausklick kann man jetzt wahlweise das aktuell selektierte Fenster oder eines von drei möglichen UIs (Desktop, Tablet oder Smartphone) starten, die gesamte Anwendung in einen lokal aufgesetzten Application-Server oder direkt in die Cloud deployen.

Mit RapidClipse kann man die eigenen Anwendungen jetzt per Mausklick über den integrierten Cloud-Dienst XDEV Cloud in die Cloud deployen, um diese beispielsweise in einer realen Produktivumgebung zu testen und gegebenenfalls später auch produktiv zu betreiben (Bild 11). Vergleichbar ist der Dienst mit der Oracle Java Cloud, die allerdings ausschließlich Oracle-Produkte unterstützt und preislich um ein Vielfaches höher liegt. Die XDEV Cloud ist eine völlig unkomplizierte Cloud-Umgebung für das Hosten von Java-Anwendungen, die nach dem Vorbild und Prinzip von Dropbox entwickelt wurde. Das Motto ist: Anmelden, hochladen, fertig. Anstelle von Bildern und Dokumenten kann man in die XDEV Cloud per Mausklick beliebige Java-Anwendungen uploaden und ausführen, in der Regel Java-Servlets.

Während man bei herkömmlichen Providern alle benötigten Server-Software-Komponenten selbst installieren, konfigurieren und warten muss, sind diese bei der XDEV Cloud bereits vorhanden und vorkonfiguriert, sodass sich der Entwickler keine Gedanken darüber machen muss, wie er einen Application-Server, Datenbankserver oder ein GIT-Repository in der Cloud zum Laufen bekommt und wie er anderen Anwendern spezielle Zugriffsrechte geben kann. Nach der Anmeldung zur XDEV Cloud erhält man Zugang zu einem sogenannten App-Container. Dabei handelt es sich um eine virtuelle Serverinstanz, auf dem bereits alle benötigten Serverkomponenten verfügbar sind (Bild 12). Der gesamte Installations- und Konfigurationsaufwand fällt damit weg.

In Kürze sollen mit Wildfly ein freier Application-Server sowie mit JBoss die Enterprise-Variante dazukommen. Auch weitere Datenbanken wie PostgreSQL und Microsoft SQL-Server sind geplant. Im Herbst soll zudem eine schnelle Java-In-Memory Datenbank bereitstehen. Über das zentrale GIT-Repository kann man den eigenen Code über das lokale GIT-Repository sicher und zentral in der Cloud ablegen, mit Kollegen oder Kunden teilen und von überall darauf zugreifen.

Beim Starten des Cloud-Deployments wird die Anwendung direkt in der Cloud gebuildet und in den Tomcat des App-Containers deployt. Nach nur etwa zehn Sekunden ist die Anwendung online. Über ein Webfrontend kann man den eigenen App-Container bei Bedarf herunterfahren und neu starten, Deployment-Prozesse anstoßen, Backups durchführen, Berechtigungen für zusätzliche Entwickler anlegen und vieles mehr.

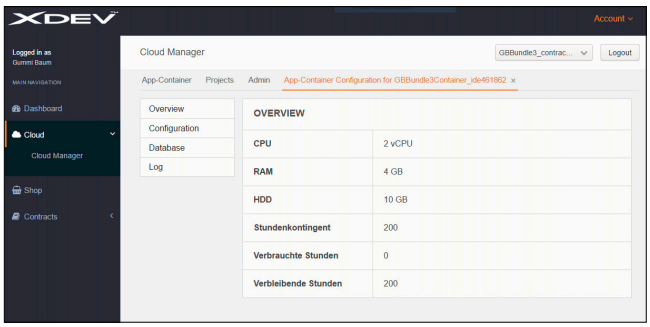
Der gesamte Dienst ist full-managed, das heißt, Backups werden täglich automatisch durchgeführt und Sicherheitsupdates vollautomatisch eingespielt. Gehostet wird der Dienst für Anwender aus Europa bei Amazon Webservices im Rechenzentrum Frankfurt. Die Nutzung ist für einen Monat kostenlos. Danach hat der Anwender die Wahl zwischen fünf Festpreis-Paketen. Im Personal-Paket für einen Entwickler soll ein App-Container ab 16,90 Euro im Monat erhältlich sein. Das Small-Team-Paket kostet 29,90 Euro monatlich. Der Service richtet sich an Entwickler, die ihre Java-Anwendungen in einer leicht bedienbaren Cloud-Umgebung testen und betreiben, sich aber nicht mit IT-Themen wie der aufwendigen Konfiguration und Optimierung von Cloud-Ressourcen sowie der Installation und Wartung von Server-Software-Komponenten herumplagen möchten (Bild 13).

Die XDEV Cloud stellt einen App-Container zur Verfügung. Alle benötigten Serverkomponenten sind vorinstalliert (Bild 12)

Leistungsfähige Mobile-Apps entwickeln

Mit Version 3 lösen die RapidClipse-Entwickler nun das Versprechen ein, mit RapidClipse leistungsfähige Industrie-4.0- und IoT-Anwendungen entwickeln und deployen zu können. Web-Apps, also Webanwendungen, die auf einem mobilen Gerät im Browser ausgeführt werden, waren mit RapidClipse bereits von Anfang an möglich. Web-Apps können jedoch nur sehr begrenzt auf System- und Gerätefunktionen zugreifen. Zugriffe auf Geo-Location, Kamera, Scanner, Bluetooth, NFC und andere Hardwarefunktionen sind jedoch Grundvoraussetzungen für Industrie-4.0-Anwendungen.

Mit Apache Cordova existiert zwar eine JavaScript-Native-Bridge, die Web-Apps genau das ermöglicht, allerdings fehlt dem von RapidClipse verwendeten UI-Framework Vaadin die nötige serverseitige Implementierung der Cordova-Features. Für Vaadin-Nutzer gab es somit bislang keine Möglichkeit, ernsthafte IoT-Anwendungen realisieren zu können. Mit RapidClipse 3 ändert sich das jetzt. Mit Hilfe eines neuen Mobile API für Vaadin kann man jetzt mit simplen Java-Methodenaufrufen von Vaadin aus direkt auf alle wichtigen System- und Gerätefunktionen von Smartphones und Tablets zugreifen und damit gängige Industrie-4.0-Anforderungen erstaunlich schnell umsetzen. Auch das bereits angekündigte Mobile Deployment ist jetzt mit dabei. Während man Web- und klassische Java-Desktop-Applikationen auf dem lokalen Rechner builden kann, werden lauffähige Android- und iOS-Apps über einen Cloud-Dienst erzeugt, was den ansonsten recht aufwendigen Build-Prozess vollständig automatisiert. Vor allem für iOS ist das ein großer Vorteil, denn um iOS-Apps builden zu können, braucht man ansonsten einen Mac. Das Einzige, was RapidClipse dem Entwickler nicht abnehmen kann, ist das Einrichten einer persönlichen Developer-ID, die man bei Google und Apple beantragen muss.

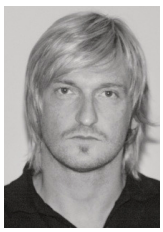


Per XDEV Cloud Manager kann man den eigenen App-Container sehr leicht konfigurieren (Bild 13)

Fazit

Wer auf Java und Eclipse umsteigen und damit HTML5-Webanwendungen, mobile Apps oder klassische Java-Desktop-Applikationen entwickeln möchte, die auf akzeptierten Java-Standard-Lösungen basieren, kommt an RapidClipse nicht vorbei. RapidClipse ist eine vollständige Eclipse-Distribution, die für Rapid Java und Cross-Platform Development vorkonfiguriert und mit zahlreichen Tools erweitert wurde.

Anders als bei Standard-Eclipse kann man RapidClipse installieren und sofort loslegen, grafische HTML5-Oberflächen und mobile Apps mit einem komfortablen GUI-Builder designen und fertige Anwendungen per Mausklick deployen. Mit der integrierten XDEV Cloud kann man eigene Anwendungen schnell und einfach in der Cloud testen sowie produktiv betreiben, ohne einen Server konfigurieren zu müssen. Mit RapidClipse steht einem schnellen Umstieg auf Java und Eclipse endlich nichts mehr im Wege. Die neue Version 3 bringt viele neue Features und setzt sogar in Sachen visuelle Anwendungsentwicklung völlig neue Maßstäbe. ■



Gerald Kammerer
ist als freier Redakteur mit Schwerpunkt Java-, Web- und App-Entwicklung tätig. Seit über zehn Jahren programmiert Gerald Kammerer intensiv mit Java.
www.redaktion-kammerer.de

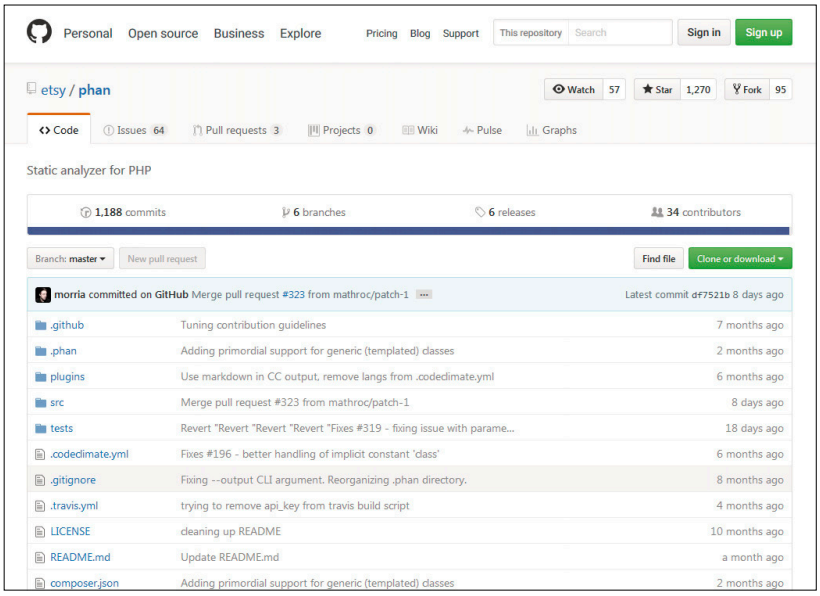
CODE-ANALYSE VON PHP-ANWENDUNGEN

Inspektor Phan

Phan ist ein Tool, das sich PHP-Programme ansieht und Schwachstellen identifiziert.

Da Phan eine statische Analyse durchführt, braucht es dabei nicht unbedingt einen Server für seine Tests. Besonders interessant: Es kann auch feststellen, ob Ihre Codebasis bereits fit für PHP 7 ist, und liefert Ihnen alle Stellen, die potenziell Ärger bereiten.

Statische Codeanalyse bedeutet, dass Ihr Programm nicht zur Laufzeit untersucht wird, sondern eine Analyse rein auf der Basis des Quellcodes erfolgt. Die Tests könnten daher theoretisch in einer beliebigen Sprache durchgeführt werden. Trotzdem verwendet Phan mit PHP genau die Sprache, deren Skripts untersucht werden sollen. Und das hat einen handfesten Grund (Bild 1).



Zentrale Anlaufstelle für das Projekt auf GitHub (Bild 1)

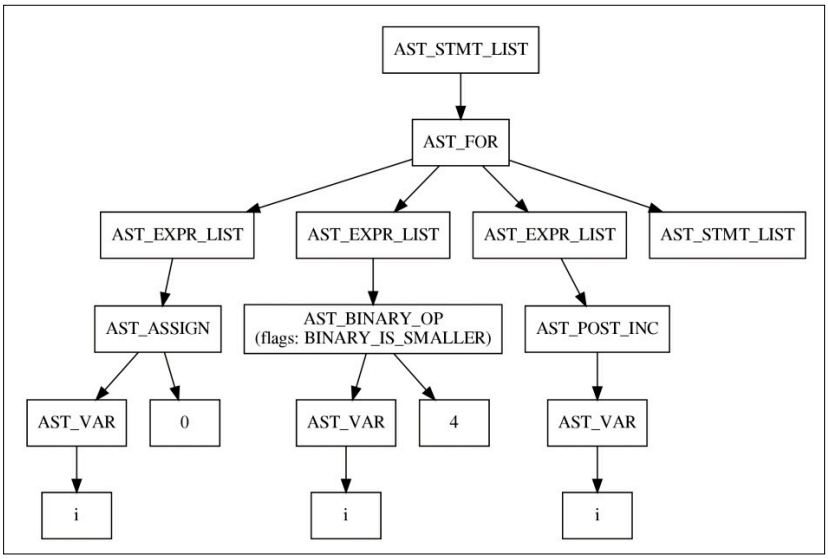
Abstract Syntax Tree

Denn seit der Version 7 zerlegt PHP vor der Ausführung jedes Programm mit Hilfe eines Abstract Syntax Tree (AST). Es wandelt also ein PHP-Skript in eine hierarchische Datenstruktur um (Bild 2).

Diese ansonsten nur intern vorliegenden Daten werden mit Hilfe der Bibliothek *php-ast* von Nikita Popov zugänglich gemacht, sodass sie über ein PHP-Skript nutzbar sind. Das Vorhandensein von AST erspart den Autoren von Phan viel Arbeit, da das Problem des Code-Parsing dadurch bereits von anderer Seite gelöst ist. Einer der beiden Hauptautoren von Phan ist übrigens der PHP-Erfinder Rasmus Lerdorf.

Klar, dass Phan durch seine Arbeitsweise einigen Themen, zum Beispiel Speicherlecks oder Kommunikationsschwierigkeiten mit APIs, nicht auf die Spur kommen kann, weil diese eben nur zur Laufzeit untersuchbar sind. Aber trotzdem kann es so wichtige Informationen sammeln. Hier die wichtigsten Erkenntnisse, die Phan über Ihr PHP-Projekt gewinnen kann:

- Abgleich, ob alle verwendeten Methoden, Funktionen, Interfaces, Traits, Properties und Variablen definiert und zugänglich sind,
- Unstimmigkeiten zwischen der Signatur einer Funktion oder Methode und ihrem Aufruf herausfinden,
- ungültige oder nicht typgerechte Nutzung von Rückgabewerten melden,
- mögliche Kompatibilitätsprobleme für den Umstieg von PHP 5 auf PHP 7 identifizieren,
- tote Codeteile und unnütze Ausdrücke ausfiltern.



So sieht die Datenstruktur aus, die PHP intern bei der Quellcode-Analyse aus einer einfachen *for*-Schleife macht (Bild 2)

Vor der Installation von Phan sollten Sie dafür sorgen, dass auf Ihrem System die Bibliotheken *php-ast* und die Unterstützung für

Sqlite eingerichtet sind. Auf Ubuntu oder Debian geben Sie dazu Folgendes ein:

```
sudo apt-get install php-ast php-sqlite3
```

Dann installieren Sie Phan selbst – etwa über Composer:

```
composer require --dev "etsy/phan:dev-master"
composer install
```

Allerdings wollen Sie vielleicht nicht den Weg über Composer nehmen, weil Sie das Test-Tool ja nicht unbedingt zum Teil Ihres Projekts machen möchten. Darum zeigt die Phan-Dokumentation einige andere Wege für die Installation, wie etwa per Docker-Container oder über Homebrew. Auch der direkte Download und die Installation via Phar sind einfach möglich:

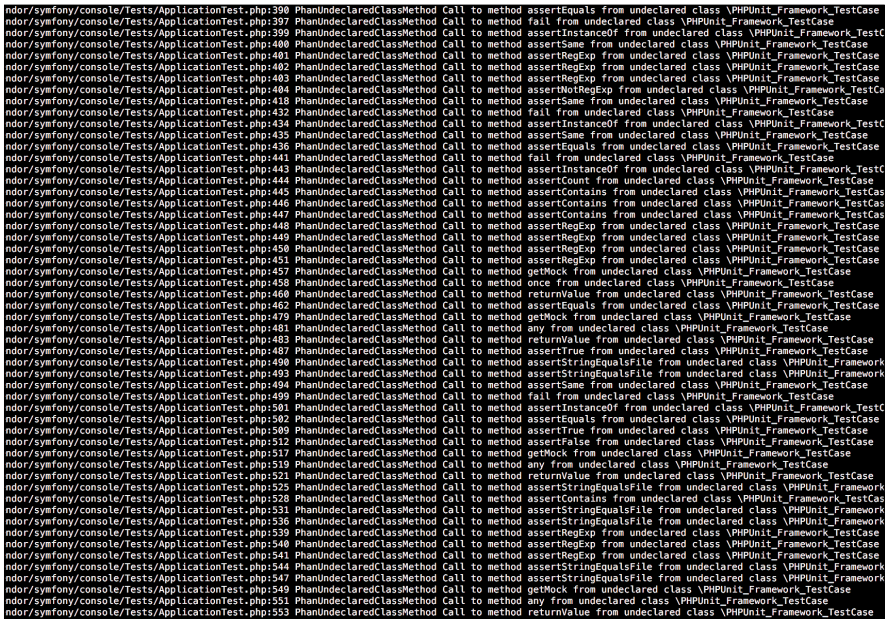
```
curl -L https://github.com/etsy/
phan/releases/download/0.6/phan.phar
-o phan.phar;
php phan.phar
```

Phan können Sie nach der Installation direkt starten und zum Beispiel dazu auffordern, alle Dateien eines Projekts durchsuchen zu lassen:

```
vendor/bin/phan /var/www/myproject
```

Hier sehen Sie die Pfadvariante für eine Composer-Installation von Phan. Wie der Weg zum Tool genau aussieht, hängt von der Art der Installation ab. Im Folgenden werden wir in den Beispielen so tun, als sei die Binary von Phan ohne weitere Pfadpräfixes erreichbar (etwa weil ein entsprechender Link für Phan in */usr/bin* vorliegt).

Als Antwort gibt Phan dann seinen Problembericht in der Form von Textmeldungen aus (Bild 3).



Phan liefert seine Fehlermeldungen standardmäßig in Textform ab. Form und Umfang können Sie aber selbst bestimmen (Bild 3)

Konfiguration einrichten

Phan kennt eine ganze Menge an Optionen für die Kommandozeile, die Sie über den Parameter *-h* erfahren können. Aber für die ernsthafte Nutzung ist ein anderer Weg vorzuziehen. Denn bei vielen Projekten werden Sie Phan immer wieder ausführen und danach die angemahnten Stellen korrigieren, bis es keine Fehler mehr wirft. Da stört es natürlich, wenn man jedes Mal die gewünschten Optionen über Aufrufparameter mitgeben muss. Darum bietet das Tool die Möglichkeit, für jedes Ihrer PHP-Projekte eine individuelle Konfiguration anzulegen.

Dazu legen Sie im Root-Verzeichnis Ihrer Webapplikation das Verzeichnis *.phan* und darin die Datei *config.php* an. Wenn Sie nun dort Phan starten, liest es die Datei ein und beachtet alle darin vorgenommenen Einstellungen. Auch auf der Kommandozeile können Sie immer noch per Parameter Einstellungen setzen, die dann höhere Priorität haben als die der *config.php*.

Die Konfiguration besteht aus einem PHP-Fragment, das ein assoziatives Array mit allen gewünschten Einstellungen festlegt. Einen beispielhaften Inhalt der Konfigurationsdatei zeigt das Listing, das Sie nicht abtippen müssen, sondern sich von der Anleitung auf Phans GitHub-Seite kopieren können. Es ist dort unter der Überschrift *Relaxed Analysis* zu finden (Bild 4).

Die gezeigte Konfiguration führt einen schnellen Test durch und geht nicht besonders in die Tiefe. Es werden auch nur die relevantesten Probleme angemahnt. Das wäre zum Beispiel eine gute Konfiguration, wenn Sie es mit einer alten Codebasis zu tun haben, wo Sie nur über die gravierendsten Probleme informiert werden möchten.

Die beiden Einstellungen *directory_list* und *exclude_analysis_directory_list* kommen im Beispiel-Listing nicht vor. Diese müssen Sie passend zu Ihrem Projekt individuell anlegen. Sie sind der Schlüssel dafür, dass Phan alle Ihre Dateien findet und korrekt bearbeitet, aber auch nur dort Alarm schlägt, wo Sie selbst eingreifen können.

Mit *directory_list* teilen Sie dem Programm mit, wo Ihre Programme zu finden sind, also zum Beispiel im Unterverzeichnis *src*. Dabei versteht Phan alle Verzeichnisangaben relativ zum Hauptverzeichnis Ihrer Anwendung:

```
'directory_list' => 'src/',
```

Der Schalter *exclude_analysis_directory_list* ist nun nicht etwa einfach eine Methode, um bestimmte Verzeichnisse ▶

von der Bearbeitung auszuschließen, sondern hat einen anderen Zweck. Er ist für Third-Party-Libraries gedacht. Die soll Phan sich zwar ansehen, um zu testen, ob Ihre Skripts deren Schnittstellen korrekt verwenden, aber nicht analysieren. Denn es hilft Ihnen ja selten weiter, wenn Sie von möglichen Problemen in externen Libraries informiert werden.

Fremdbibliotheken

Haben Sie also zum Beispiel Fremdbibliotheken im Unterverzeichnis *vendor* abgelegt, dann setzen Sie die Einstellung so:

```
'exclude_analysis_directory_list' => 'vendor/',
```

Wenn Sie alle Schalter so gesetzt haben, dass sie für Ihr Projekt passen, starten Sie Phan:

```
phan --progress-bar
```

Der Parameter *--progress-bar* (kurz *-p)* sorgt dafür, dass Sie einen Verlaufsbalken angezeigt bekommen, der Ihnen eine Abschätzung der gesamten Bearbeitungszeit ermöglicht. Zusätzlich wird die Datenmenge angezeigt, die Phan bearbeitet und analysiert.

Explizite Dateiliste vorgeben

Normalerweise sucht sich Phan ja die PHP-Dateien aus den in *directory_list* festgelegten oder in der Kommandozeile genannten Verzeichnissen selbstständig heraus. In manchen Fällen ist es aber vielleicht angebracht, dass Sie dem Tool eine Liste mit Ihren Skripts übergeben. Dazu dient der Parameter *-f*, dem ein Dateiname folgen muss:

```
phan -f files.txt
```

Wenn Ihr Projekt aus vielen einzelnen Programmdateien besteht, ist die manuelle Erzeugung dieser Dateiliste unsinnig. Stattdessen könnten Sie zuerst über ein passendes *grep*-Kommando alle Dateien herausfinden, in denen die Zeichenkette *<?php* vorkommt:

```
grep -R -l "<?php" ./myproject/* > myphpfiles.txt
```

Dann bearbeiten Sie die Dateiliste *myphpfiles.txt* mit einem Editor und löschen alle Einträge heraus, die das Tool nicht untersuchen soll. Nun können Sie Phan mit Ihrer individuellen Liste von Dateien starten:

```
phan -f myphpfiles.txt
```

Vielleicht meldet Phan ganz viele Fundstellen eines bestimmten Typus, der Ihnen gar kein Kopfzerbrechen bereitet. Das ist dann ärgerlich, weil die schiere Menge die für Sie wichtigeren anderen Probleme in den Hintergrund drängt.

Um diesem Fall zu begegnen, kennt die Konfiguration den Schlüssel *suppress_issue_types*. Dem können Sie ein Array an Phan-Errorcodes mitgeben, die Sie nicht mehr ausgegeben haben möchten. Um beispielsweise keine Meldungen über undeklarierte Methoden und Funktionen zu erhalten, fügen Sie in der *config.php* diese Zeile ein:

```
'suppress_issue_types' => ['PhanUndeclaredMethod'],
```

Ein alternativer Ansatz ist die Nutzung von Annotations. Die verwendet Phan einerseits dazu, um mehr über erwartete Variablentypen zu erfahren, wie in diesem Beispiel:

```
/** @var integer */
$counter = 0;
```

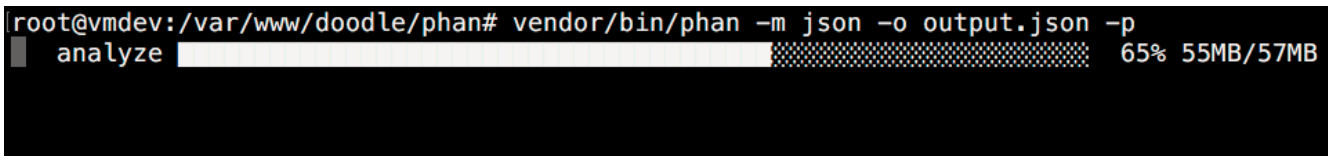
Dabei kennt Phan auch die anderen in der PHP-Welt üblichen Annotation-Codes wie *@param* oder *@return*. Das Tool führt aber auch den neuen Annotation-Code *@suppress* ein:

```
class ClassA {
    /** @suppress
    PhanUndeclaredClassMethod */
    function funcA() {
      ClassB::funcB();
    }
}
```

Die Anmerkung würde nun dazu führen, dass der Aufruf von *ClassB::funcB()* nicht zu einer Fehlermeldung führt, wenn *funcB()* nicht deklariert wurde. Allerdings müssen Sie bei der Verwendung von solchen Doc-Blocks in Phan besonders auf die Syntax achtgeben. So übergeht Phan einfach Annotations, die die einzeilige Syntax für Kommentare verwenden. Die folgende Zeile bleibt also wirkungslos:

```php
<?php
/**
 * This configuration will be read and overlaid on top of the
 * default configuration. Command line arguments will be applied
 * after this file is read.
 *
 * @see src/Phan/Config.php
 * See Config for all configurable options.
 */
return [
    // Backwards Compatibility Checking. This is slow
    // and expensive, but you should consider running
    // it before upgrading your version of PHP to a
    // new version that has backward compatibility
    // breaks.
    'backward_compatibility_checks' => false,

    // Run a quick version of checks that takes less
    // time at the cost of not running as thorough
    // an analysis. You should consider setting this
    // to true only when you wish you had more issues
    // to fix in your code base.
    'quick_mode' => true,

    // If enabled, check all methods that override a
    // parent method to make sure its signature is
    // compatible with the parent's. This check
    // can add quite a bit of time to the analysis.
    'analyze_signature_compatibility' => false,

    // The minimum severity level to report on. This can be
    // set to Issue::SEVERITY_LOW, Issue::SEVERITY_NORMAL or
    // Issue::SEVERITY_CRITICAL. Setting it to only
    // critical issues is a good place to start on a big
    // sloppy mature code base.
    'minimum_severity' => 10,

    // If true, missing properties will be created when
    // they are first seen. If false, we'll report an
    // error message if there is an attempt to write
    // to a class property that wasn't explicitly
    // defined.
    'allow_missing_properties' => true,

    // Allow null to be cast as any type and for any
    // type to be cast to null. Setting this to false
    // will cut down on false positives.
    'null_casts_as_any_type' => true,

    // If enabled, scalars (int, float, bool, string, null)
    // are treated as if they can cast to each other.
    'scalar_implicit_cast' => true,

    // If true, seemingly undeclared variables in the global
    // scope will be ignored. This is useful for projects
    // with complicated cross-file globals that you have no
    // hope of fixing.
    'ignore_undeclared_variables_in_global_scope' => true,

    // Add any issue types (such as 'PhanUndeclaredMethod')
    // to this black-list to inhibit them from being reported.
    'suppress_issue_types' => [
        // 'PhanUndeclaredMethod',
    ],

    // If empty, no filter against issues types will be applied.
    // If this white-list is non-empty, only issues within the lis
    // will be emitted by Phan.
    'whitelist_issue_types' => [
        // 'PhanAccessMethodPrivate',
    ],
];
```

Mit einer Konfigurationsdatei – hier ein Beispiel mit relativ laxen Prüfungen – legen Sie genau fest, wie Phan seine Analyse durchführen soll (Bild 4)



Bei der Verwendung einer separaten Datei für die Programmausgabe sollten Sie Phar mit der Option für den Verlaufsbalken starten (Bild 5)

```
// @suppress PhanUndeclaredClassMethod
```

Außerdem ist die Positionierung im Code entscheidend. Ein *@suppress* bezieht sich immer auf den nächsten Sichtbarkeitsbereich. Sehen Sie sich dazu diese Variante des obigen Schnipsels an:

```
/** @suppress PhanUndeclaredClassMethod */
class ClassA {
  function funcA() {
    ClassB::funcB();
  }
}
```

Hier bezieht sich die Anmerkung auf die Klasse, unterdrückt also den Fehler beim Aufruf von *funcB()* nicht, weil dieser ja erst im Scope von *funcA()* erfolgt.

Aus Sicherheitsgründen und vor allem auch wegen der schnelleren Ausführungsgeschwindigkeit wollen Sie natürlich auch ältere Anwendungen in den Genuss eines Updates auf PHP 7 kommen lassen. Auf die dabei auftretenden Stolpersteine kann Phan Sie hinweisen.

Dazu fügen Sie in der *config.php* dem Konfigurations-Array die folgenden drei Elemente hinzu:

```
'backward_compatibility_checks' => true,
'should_visit_all_nodes' => true,
'whitelist_issue_types' => ['PhanCompatiblePHP7'],
```

Mit der ersten Zeile schalten Sie den Prüfer für die Kompatibilität ein. Damit gibt Ihnen Phan alle Stellen aus, die durch Änderungen in der Bewertung von Ausdrücken, beim Variablen-Handling oder durch andere Neuerungen potenziell unerwünschte Auswirkungen auf Ihre Anwendung haben, sobald Sie auf PHP 7 umstellen. Die zweite Zeile sorgt dafür, dass Phan tiefer in den AST-Nodes sucht, als mit den Default-Einstellungen festgelegt. Der letzte Eintrag wiederum ist quasi genau das Gegenteil von *suppress_issue_types*. Denn wenn hier mindestens ein Phan-Fehlercode genannt wird, gibt das Tool nur noch die darunter fallenden Probleme aus und blendet alle anderen aus.

Wenn Sie Phan überhaupt nur dazu nutzen wollen, schnell die Kompatibilitätsprobleme grob zu identifizieren, brauchen Sie übrigens nicht unbedingt mit einer eigenen Konfigurationsdatei zu arbeiten. Sie können auch einfach an der Kommandozeile mit dem Parameter *-b* oder der langen Variante *--backward-compatibility-checks* arbeiten. Dann erscheinen im Gegensatz zur Variante per *config.php* allerdings auch andere gefundene Probleme, und eventuell gehen dem Tool wegen der geringeren Suchtiefe manche Dinge durch die Lappen. Sollte Sie damit leben können, dann genügt ein Aufruf von Phan in der folgenden Art:

```
phan -b /var/www/myproject
```

Standardmäßig gibt Phan seine Erkenntnisse einfach als zeilenweise Textmeldungen zurück. Wenn Sie das Programm allerdings als Baustein einsetzen möchten – etwa in einer Automatisierungslösung fürs Deployment –, dann ist eventuell ein anderes Format günstiger (Bild 5).

Dazu kennt Phan die Alternativen CSV, JSON sowie CheckStyle und CodeClimate. Die beiden letztgenannten sind proprietäre Formate der gleichnamigen Tools für die Qualitätssicherung von Programmcode. Um auf eines der anderen Formate umzuschalten, nutzen Sie den Programmparameter *--output-mode* mit einem der Formatnamen in Kleinschreibweise. Möchten Sie direkt eine Ausgabedatei angeben, statt die Meldungen direkt in der Konsole zu erhalten, setzen Sie den Parameter *-o* ein. ■

Links zum Thema

- Zentrale Anlaufstelle für das Projekt auf GitHub
  https://github.com/etsy/phan
- Installation von Phan über verschiedene Methoden
  https://github.com/etsy/phan#getting-phan-running
- Erklärungen aller Änderungen in PHP 7, die mögliche Probleme bei einem Update nach sich ziehen
  http://php.net/manual/en/migration70.incompatible.php
- Repository von php-ast
  https://github.com/nikic/php-ast

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

Auswählen und freistellen

Photoshop liefert dem Designer die notwendigen Tools zum Auswählen und Freistellen.

Das Freistellen ist ein häufig verwendeter Arbeitsvorgang in der Bildbearbeitung. Hierbei werden die Bereiche in einem Foto gewählt, die sichtbar sein sollen. Der Rest der Bildinformationen wird entweder gelöscht oder maskiert, also unsichtbar gemacht.

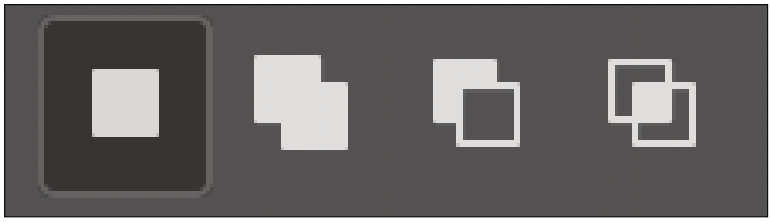
Das gewählte Motiv, der sogenannte Freisteller, kann auf einen anderen Hintergrund gesetzt werden: Bei Produktfotos handelt es sich häufig um eine einfarbige Fläche, die sich dem allgemeinen Hintergrund der darstellenden Seite anpasst. Zudem eignen sich Freisteller als Bestandteile einer Collage oder Fotomontage. Einen weiteren Anwendungsbereich einer solchen Objektauswahl stellen zusätzliche Bildbearbeitungen des Motivs oder des Hintergrunds dar, etwa Scharf- oder Weichzeichnen oder verschiedene Farb- und Helligkeitskorrekturen.

Um ein Objekt freizustellen, bedarf es zunächst einer Auswahl. Diese kann mit unterschiedlichen Werkzeugen oder auch Funktionen erstellt werden. Hierzu zählen die verschiedenen Auswahlwerkzeuge, die sich in den einzelnen Bildbearbeitungslösungen mehr oder weniger gleichen. Besonders viele Tools, die zudem fein einstellbar sind, liefert jedoch Adobe Photoshop CC 2015.

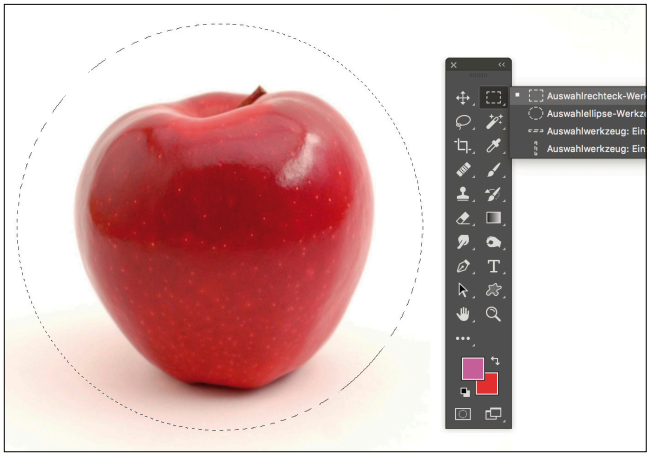
Auswahlwerkzeuge in Photoshop CC 2015

Alle Werkzeuge zum Selektieren eines Objekts befinden sich bei Photoshop im oberen Bereich der Werkzeugleiste. Dazu liefert die im oberen Bereich der Arbeitsfläche angesiedelte Optionsleiste weitere Einstellungen, etwa die Auswahloptionen: Neben einer neuen Auswahl lassen sich Bereiche aus einer bestehenden Auswahl ausschneiden oder hinzufügen. Ist die letzte Schaltfläche gewählt, bildet Photoshop eine Schnittmenge der gewählten Flächen (Bild 1).

In der Werkzeugleiste, unter den geometrischen Auswahlwerkzeugen, ist zunächst das Auswahlrechteck-Werkzeug zu finden. Hinter ihm verbergen sich drei weitere Möglichkeiten zum Wählen von elliptischen oder kreisrunden Flächen, einer einzelnen Zeile oder einer Spalte. Die jeweilige



Eine bestehende Auswahl kann mit einer neuen Auswahl in unterschiedlicher Art und Weise kombiniert werden (Bild 1)



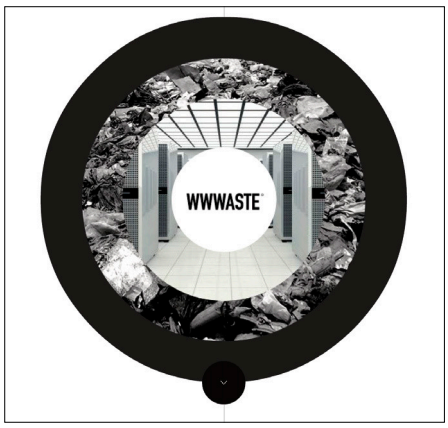
Die vier Auswahlwerkzeuge von Photoshop eignen sich lediglich zum Ausschneiden geometrischer Flächen (Bild 2)

Auswahl wird direkt auf der Arbeitsfläche aufgezogen. Auswahlen dieser Art können zwar über das Menü *Auswahl, Auswahl verändern* oder *Auswahl, Auswahl transformieren* in ihrer Größe und den Proportionen geändert werden, eine exakte Anpassung an ein Bildelement, das keine geometrische Form aufweist, ist jedoch nicht möglich, wie Bild 2 zeigt.

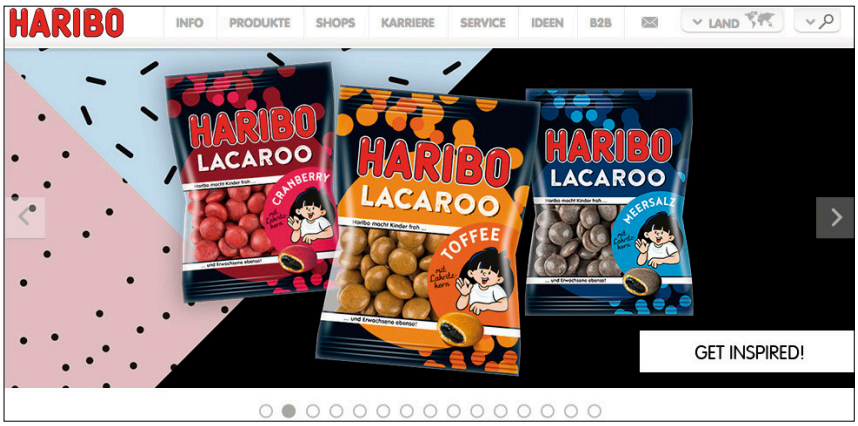
Eine individuelle Form kann mit diesen Mitteln nicht gewählt werden. Geometrische Auswahlen sind dennoch für verschiedene Aufgaben sehr praktisch: So etwa für einen kreisrunden Bildausschnitt, der in einem Layout mit anderen Elementen kombiniert werden soll. Ein gutes Beispiel liefert die Website wwwaste.fr (Bild 3), auf der gleich zwei kreisrunde Bildausschnitte in der Bildmitte des Startscreens hintereinander eingeblendet werden, bis sie übereinanderliegen.

Für exakte Formen sind die drei Lasso-Werkzeuge zuständig. Mit dem obersten, dem einfachen Lasso-Werkzeug, kann eine Auswahl wie mit einem Pinsel mit gedrückter Maustaste um das zu wählende Element gezogen werden. Selbst bei einer einfachen Form, wie bei dem Apfel im Beispiel, ist es jedoch kaum möglich. eine einigermaßen zufriedenstellende Auswahl zu erzeugen – schon gar nicht mit der Maus, eher noch mit einem Grafiktablett.

Hier liefert das Magnetisches-Lasso-Werkzeug ein besseres Ergebnis: Wird die Maus entlang einer Bildkante bewegt, sucht Photoshop selbstständig die Grenzen. Zu sehen ist dann eine dünne Linie mit mehreren Befestigungspunkten, die ebenfalls per Mausklick manuell gesetzt werden können. Für ein gutes Gelingen sind neben den unterschiedlichen Auswahloptionen weitere Einstellun-



Auf wwwaste.fr kommen kreisrund ausgewählte Bildausschnitte zum Einsatz (Bild 3)



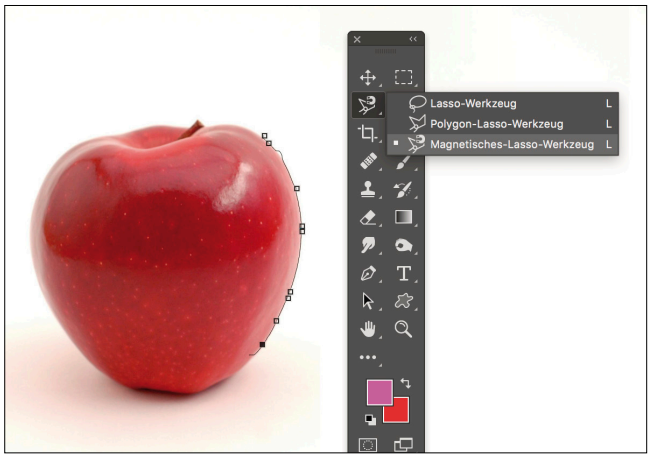
Haribo bildet die verschiedenen Produkte auf zusammengesetzten, gerade abgeschnittenen Musterflächen ab (Bild 5)

gen in der Optionsleiste wichtig: Die Breite definiert den Abstand, in dem die Kante erkannt wird, der Kontrast regelt die Empfindlichkeit, und die Frequenz ist zuständig für Anzahl und Abstand der einzelnen Befestigungspunkte zueinander. Hier ist es jedoch entscheidend, dass die Kanten scharf abgebildet sind und sich das Motiv gut vom Hintergrund unterscheidet (Bild 4). Bei relativ einfachen Formen ist dieses Tool eine recht gute Wahl, für komplexere Elemente eignet es sich jedoch weniger, da man hier leicht den Faden verliert. Dadurch können Bereiche aus dem Hintergrund mit in die Auswahl geraten, die dann in mühsamer Kleinarbeit wieder daraus entfernt werden müssen.

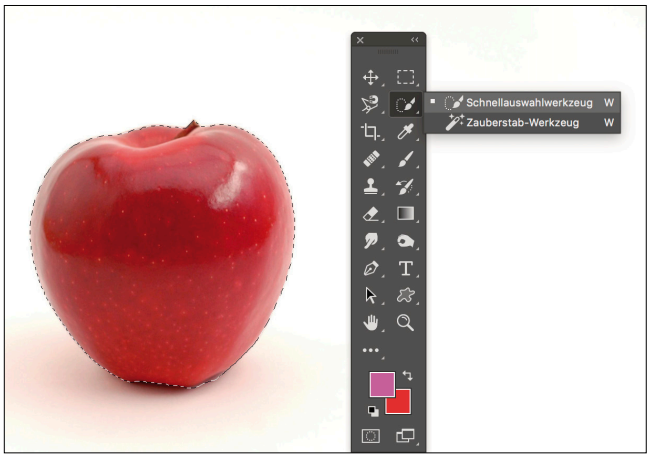
Mit dem Polygon-Lasso-Werkzeug entstehen Auswahlflächen, die ohne kurvige Segmente auskommen. Dabei legt jeder Mausklick einen Eckpunkt fest, per Doppelklick schließt sich die Auswahl. Neben dem Erstellen von Bildausschnitten können auch Musterflächen per Auswahl begrenzt werden, die scharfe Kanten und gerade Ränder zeigen. Haribo etwa bedient sich dieser Technik auf der Startseite: Hier werden die Produkte auf unterschiedlichen Hintergründen vorgestellt, die aus mehreren geradlinig aneinandergrenzenden Flächen bestehen. Dabei handelt es sich teils um ausgeschnittene Fotos, um reine Farbflächen oder auch um verschiedene Muster, wie das Beispiel im Bild 5 zeigt. Die auf dem Hintergrund liegenden Produkttüten zeigen hingegen nicht nur gerade Kanten; sie könnten mit dem Magnetisches-Lasso-Werkzeug ausgewählt worden sein.

Zauberstab und Schnellauswahl dienen ebenfalls dazu, einfache Objekte auszuwählen – und können das meist besser als das Polygon-Lasso-Werkzeug.

Liegt das Motiv auf einer mehr oder weniger reinen Farbfläche, kann diese mit einem Mausklick mit dem Zauberstab-Werkzeug ausgewählt werden. Soll statt des Hintergrunds das Motiv gewählt sein, wird die Auswahl über das Auswahl-Menü umgekehrt. Zwei Optionen sind für einen schnellen Erfolg wichtig: der Aufnahme-Bereich und die Toleranz. Der Aufnahme-Bereich definiert die Fläche, die Photoshop für die eigentliche Auswahl als Referenz-Farbton nimmt. Soll beispielsweise ein blauer Himmel gewählt werden und zeigt das Bild recht starkes Farbrauschen, so zeigen die nebeneinanderliegenden Pixel verschiedene Blautöne. In diesem Fall eignet sich etwa ein Wert von 5 x 5 Pixel Durchschnitt; Photoshop errechnet dann den Mittelwert der Pixel, die in diesem per Mausklick gewählten Bereich liegen. Die Toleranz ▶



Das Magnetisches-Lasso-Werkzeug sucht eigenständig nach Kanten im Bild (Bild 4)



Der Zauberstab wählt mehr oder weniger reine Farbflächen per Mausklick aus (Bild 6)

bestimmt, inwieweit auch Farb- und Tonwerte, die von der gewählten Farbe abweichen, mit in die Auswahl übernommen werden (Bild 6).

Befinden sich auf einem einfarbigen Hintergrund mehrere Objekte, von denen nur eines gewählt werden soll, ist der Einsatz des Schnellauswahlwerkzeugs sinnvoll. Auf Bild 7 befinden sich beispielsweise drei Äpfel, es soll jedoch nur der rote zuoberst ausgeschnitten werden. Mit dem Schnellauswahlwerkzeug wird mit kurzen Mausstrichen das Objekt vollständig gewählt, Photoshop erkennt dabei die Ränder.

Der passende Hintergrund

Ist bereits vor der Aufnahme des Motivs klar, dass dieses freigestellt werden soll, gilt es, bereits bei der Aufnahme für den passenden Hintergrund zu sorgen.

Am besten ist natürlich eine Aufnahme, die im Studio vor weißem Hintergrund gemacht wurde. Dabei kann mit professioneller Beleuchtung das Objekt so ausgeleuchtet werden, dass möglichst wenig und nur sehr schwache Schatten im Bild liegen – die Auswahl mit Zauberstab und Co. sollte bei diesen Fotos mit den passenden Einstellungen nun ohne viel Aufwand funktionieren.

Ist keine Studioaufnahme möglich, sollte sich der Hintergrund in Farbe und Helligkeit möglichst stark vom Objekt abheben. Zudem punkten hier gestochen scharfe Aufnahmen: Verwackelte Kanten lassen sich nur schwer fassen. Dazu bleiben in der Regel Farbreste aus dem Hintergrund in den unscharfen und damit halbtransparenten Bereichen hängen; das später ausgeschnittene Objekt wird sich so kaum in eine neue Umgebung stimmig einfügen lassen. Ein Beispiel dazu zeigt Bild 8: Dieser Holzstuhl kann kaum mit wenigen Mausklicks ausgewählt werden. Bei einem ersten Versuch mit dem Schnellauswahlwerkzeug gelangen Details der Körbe im Hintergrund mit in die Auswahl. Zudem erkennt das Werkzeug rechts im Bild die Grenze zwischen der Vorderseite des linken vorderen Stuhlbeins und dem Bodenbelag nicht. Zwar könnte hier in mühsamer Kleinarbeit mit den unterschiedlichen Werkzeugen eine halbwegs passende Auswahl erstellt werden; da der Stuhl jedoch sehr unscharf abgebildet ist, ist

Die Auswahl des Stuhls vor dem unruhigen Hintergrund kann nicht zufriedenstellend funktionieren (Bild 8)

das freigestellte Objekt für einen späteren Gebrauch mehr oder weniger unbrauchbar.

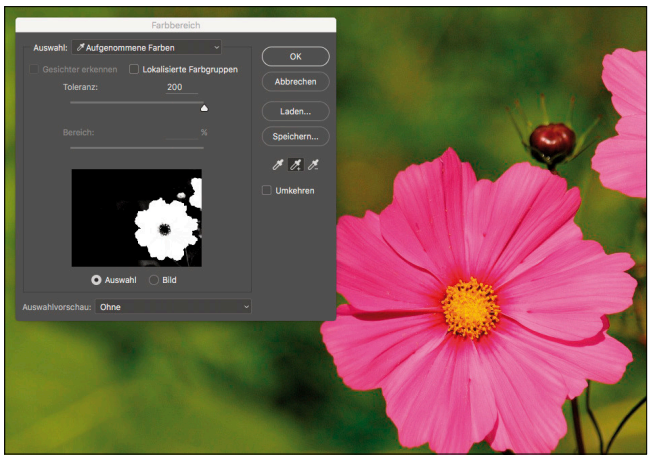
Zum Vergleich: Baur Versand wirbt auf einer Landingpage für neue Möbel und hat dazu einen grünen Stuhl in Szene gesetzt. Diese Seitengestaltung funktioniert nur dadurch, dass der Stuhl gestochen scharf abgebildet und ebenso professionell freigestellt worden ist (Bild 9).

Weitere Auswahlfunktionen

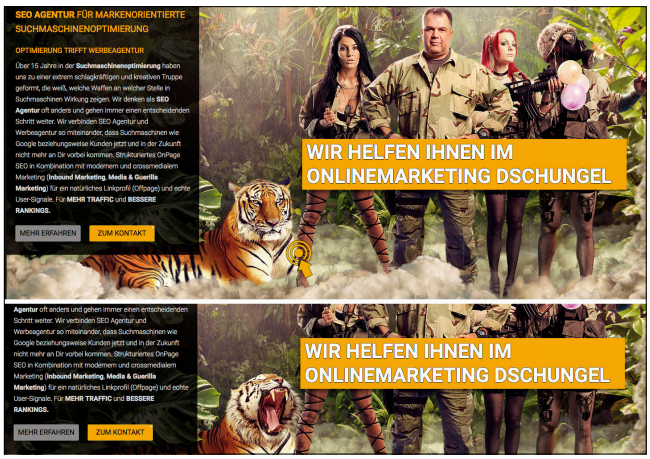
Weitere Auswahlmöglichkeiten sind im Menü *Auswahl* zu finden, etwa die Funktionen *Farbbereich* und *Fokusbereich*. Zudem bietet Photoshop seinen Anwendern die überarbeitete Funktion *Auswählen und maskieren*. *Farbbereich* basiert – wie der Zauberstab – auf der Wahl eines Farbwertes. Über *Auswahl, Farbbereich* öffnet Photoshop einen Dialog, in dem

Das Schnellauswahlwerkzeug erkennt zusammenhängende Flächen im Bild (Bild 7)

Beispiel Baur.de: Erst ein professionell ausgewähltes Objekt entfaltet innerhalb der Produktfotografie seine Wirkung (Bild 9)



Die Funktion Farbauswahl funktioniert ähnlich wie der Zauberstab, liefert jedoch weitere Optionen (Bild 10)

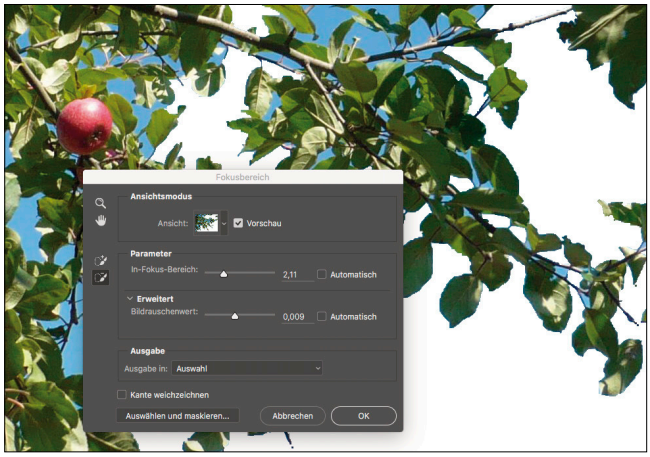


Eine Animation des Tigers zeigt dessen Mähne vor einer anderen Stelle im Hintergrund (Bild 12)

zunächst per Pipette eine Farbe gewählt wird. Auch hier bestimmt die Toleranz, welche Abstufungen der gewählten Farbe in die Auswahl gelangen. Zwei weitere Pipetten dienen dazu, Farbbereiche aus der Auswahl zu löschen oder hinzuzufügen. Weiter kann die Auswahl umgekehrt werden, was recht praktisch ist, wenn etwa ein einfarbiger Hintergrund gewählt wurde (Bild 10).

Bei dem Fokusbereich versucht Photoshop mehr oder weniger erfolgreich, die Tiefenschärfe im Bild zu analysieren, um dann die Bereiche aus der Auswahl auszuschließen, die im Hintergrund liegen. Zum Einstellen gibt es die Parameter *In-Fokus-Bereich* und unter *Erweitert* den *Bildrauschwert*.

Bei Bildern mit einem gestochen scharf abgebildeten Objekt und einem durch Tiefenunschärfe entsprechend weichgezeichneten Hintergrund erstellt das Programm eine recht gute Auswahl, die jedoch – je nach Komplexität des Motivs – mit anderen Mitteln verbessert werden muss. So zeigt etwa Bild 11, dass Photoshop den blauen Himmel hinter dem Apfelbaum durchaus als Hintergrund erkannt hat, schließt jedoch viele kleine Details davon nicht ein. Hier wäre wahrscheinlich eine Auswahl mit dem Zauberstab einfacher.



Fokusbereich erkennt die Tiefenschärfe im Bild – mehr oder weniger erfolgreich (Bild 11)

Glatte Kanten – ob gerade oder gewellt – sind mit etwas Aufwand mit allen bisher vorgestellten Mitteln in der Regel gut freizustellen. Eine echte Herausforderung bilden jedoch Motive, deren Kanten stark ausgefranst und dabei teilweise transparent sind, etwa Haare oder Fell. Gerade wenn solche Objekte vor einen neuen Hintergrund gestellt werden sollen, muss die Auswahl stimmen.

Als Beispiel sei Bild 12 genannt: Hier zeigt die Startseite der SEO-Agentur Sumago eine witzige Animation des Tigers im unteren Bereich: Per Klick auf die Schaltfläche bewegt er zum Brüllen seinen Kopf – die Mähne ist nun an anderer Stelle vor dem Hintergrund zu sehen.

Für solche komplexen Auswahlen hat Adobe die ehemalige Option *Kanten verbessern* mit dem aktuellen Update zu dem Dialog *Auswählen und maskieren* erweitert. Um diese anzuwenden, muss zunächst eine möglichst gute Auswahl des Motivs stehen. Ist dann ein Auswahlwerkzeug aktiv, steht in der Optionsleiste die Funktion *Auswählen und maskieren* bereit. Per Smartradius wird nun ein Bereich um die Kante definiert, in dem sich halbtransparente Details befinden, also etwa Haare.

Diese Bereiche können mit den Werkzeugen am linken Rand der Arbeitsfläche weiter ausgearbeitet werden. So lassen sich innerhalb einer Auswahl scharfe Ränder ebenso wie eine wehende Mähne definieren. Um Farbsäume zu vermeiden, die etwa durch eine grüne Wiese im Hintergrund entstehen, muss in den Ausgabeeinstellungen die Option *Farbe dekontaminieren* aktiviert sein. ■

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.

Foto: Shutterstock / phipatbig

PREDICTIVE ANALYTICS

Fundierte Entscheidungen

Mit modernen Vorhersagemethoden gelingen Unternehmen treffendere Prognosen. Predictive-Analytics-Lösungen führen dabei zu fundierten Entscheidungen.

Ganz gleich ob im Einkauf, im Vertrieb, im Marketing oder in der Chefetage: In jedem Unternehmen werden ständig Entscheidungen getroffen, die auf Annahmen über die Zukunft basieren. »Um Entscheidungs- und Handlungsalternativen entwickeln zu können, brauchen wir den Blick nach vorn«, sagt Hans Wieser, Business Lead Data Platform bei Microsoft Deutschland. Neben einfachen statistischen Verfahren wie Extrapolation oder Regression und dem viel zitierten »Bauchgefühl« werden dabei mit Hilfe der IT zunehmend fortgeschrittene Analysen eingesetzt, die dank sogenannter Predictive-Analytics-Software (PA) auch bei großen Datenmengen und komplexen Zusammenhängen zuverlässige Vorhersagen ermöglichen. Dieser Ansatz stoße auf großes Interesse, so Wieser: »Das ist in fast jeder Branche derzeit ein Fokusthema.«

Die Marktforscher sind derselben Meinung und prognostizieren für Predictive Analytics ein starkes Wachstum. Laut Technavio steigen die Umsätze bis 2019 weltweit um durchschnittlich 25 Prozent pro Jahr, Research and Markets und Micromarket Monitor halten sogar Wachstumsraten von über 30 Prozent für möglich. Die Angaben und Prognosen über das Gesamtvolumen des PA-Markts liegen zwischen 2,3 Milliarden Dollar im Jahr 2019 bei Micromarket Monitor und 5,24 Milliarden Dollar im Jahr 2018 bei Research and Markets.

Softwarehersteller rüsten auf

Die Markterwartungen wecken auch Begehrlichkeiten bei den Konzernen, die selbst nicht oder nicht genügend eigene Expertise im Bereich fortgeschrittener Analysen haben. Bereits 2007 kaufte Tibco Software den Business-Intelligence-Spezialisten Spotfire, 2009 verleibte sich IBM die Statistikfirma SPSS ein, 2014 übernahm Dell StatSoft und Anfang 2015 schluckte OpenText den BI-Anbieter Actuate. Auch Microsoft ist auf Einkaufstour und hat im vergangenen Frühjahr die Übernahme von Revolution Analytics abgeschlossen, einem Unternehmen, das mit Revolution R eine Distribution des für fortgeschrittene Analysen häufig verwendeten Programmpakets R anbietet (Tabelle 1).

Das steigende Interesse hat gute Gründe, findet Heleen Snelting, Data Science Manager EMEA bei Tibco Software:

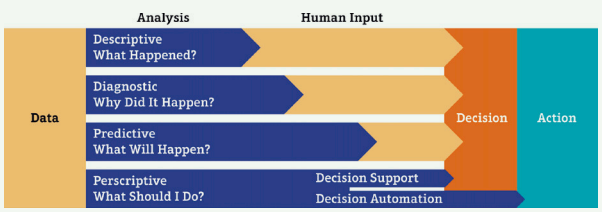
Tabelle 1: Anbieter von Predictive-Analytics-Lösungen (Auswahl)

Hersteller	Lösungen	Internet
Blue Yonder	Blue Yonder Platform (Predictive Applications as a Service), Forward Demand, Forward Pricing	www.blue-yonder.com/produkte.html
Dell StatSoft	Statistica	www.statsoft.de
IBM SPSS	Statistics, Modeler	www-01.ibm.com/software/de/analytics/spss
Information Builders	WebFocus Rstat, Smart-Edge	www.informationbuilders.de/solutions/predictive-analytics
Microsoft	SQL Server, Azure, Revolution Analytics Revolution R	www.microsoft.com/de-de/server-cloud, www.revolutionanalytics.com
OpenText	Big Data Analytics (Actuate BIRT)	http://birtanalytics.actuate.com/predictive-analytics
Pentaho	Big Data, Business Analytics	www.pentaho.de
RapidMiner	RapidMiner Studio, Server, Radoop, Cloud	https://rapidminer.com
SAP	Predictive Analytics	http://go.sap.com/germany/solution/platform-technology/predictive-analytics.html
SAS	Enterprise Miner, Factory Miner, Text Miner, Forecast Server und andere	www.sas.com/de_de/software/analytics.html
Skytree	Skytree Machine Learning Software	www.skytree.net/products
Teradata	Integrated Marketing Cloud, Warehouse Miner, Geospatial und andere	http://marketing.teradata.com, www.teradata.de
Tibco	Spotfire	http://spotfire.tibco.com/de/solutions/technology/predictive-analytics

»Predictive Analytics bietet Unternehmen einen signifikanten Wettbewerbsvorteil, weil sie schnellere und fundierte Entscheidungen treffen können. Unternehmen sind dadurch in der Lage, Chancen besser zu nutzen und Risiken vorzeitig zu erkennen und zu umgehen.« Die dabei angewandten Verfahren wie multivariate Statistik, neuronale Netze oder maschinelles Lernen sind nicht neu. Sie werden schon seit vielen Jahren in der Forschung, aber auch in der Wirtschaft eingesetzt. Neu in Predictive Analytics sind vor allem vier Aspekte:

- **Demokratisierung der Verfahren:** Wie im Bereich Business Intelligence werden auch die Prognosemodelle von Predictive Analytics immer leichter zugänglich. Grafische Bedienoberflächen ersetzen kommandozeilenbasierte Programmierschnittstellen und erlauben es, Analysen per Drag and Drop zusammenzustellen. Cloud-Angebote erleichtern außerdem den Zugang zu fortgeschrittenen Analysemethoden, ohne vorab investieren und komplexe Systeme installieren zu müssen. »Insbesondere im Bereich Labor sehe ich aufgrund der Pay-as-you-go-Angebote, der häufigen Aktualisierungen und der Möglichkeiten zur Skalierung Cloud-Werkzeuge auf dem Vormarsch«, sagt Artus Krohn-Grimberghe, Chief Data Scientist bei der pmOne Analytics GmbH, einer 2015 gegründeten Tochter der pmOne AG, die neben Beratungsleistungen auch eigene Algorithmen für fortgeschrittene Analysen entwickeln und anbieten will.
- **Auswertung großer Datenmengen:** Durch Big-Data-Analyseverfahren lassen sich Vorhersagen auf Basis großer Mengen unstrukturierter Daten treffen. »Durch die aktuelle Popularität des Begriffs Big Data ist das Interesse in allen Branchen stark gewachsen, Daten zu nutzen, um wertschöpfende Informationen zu generieren«, sagt Nicole Tschauder, Business Expert Analytics bei SAS Deutschland.
- **Vernetzung verschiedenster Informationen:** »Wo man früher eine Datenbank eingesetzt hat, nutzt man heute viele verschiedene Datenquellen«, erläutert Jakob Rehermann, Geschäftsführer des Berliner Software-Unternehmens Datapine, das seit vier Jahren eine einfach zu bedienende Software-as-a-Service-basierte Business-Intelligence-Lösung für kleine und mittlere Unternehmen entwickelt und ab April auch ein Modul zur Erstellung von Vorhersagen anbieten will.
- **Interne und externe Quellen:** Während klassische Prognoseverfahren hauptsächlich auf intern erhobenen Daten wie Absatzzahlen oder Einkaufspreisen beruhen, setzen ▶

So sehen die Gartner-Analysten Predictive Analytics



Mehrere Stufen führen von rein deskriptiven Methoden zu automatischen Reaktionen auf Vorhersagen

web & mobile developer 11/2016 Quelle: Gartner

heute viele Anwendungsfälle auf eine Kombination interner Informationen mit externen Angeboten wie Wettervorhersagen, demografischen Entwicklungen, Verkehrsströmen oder auch Social-Media-Daten. »Je mehr Informationsquellen ich habe, um so runder wird mein Bild vom Kunden«, sagt Dominik Claßen, Sales Engineering Director EMEA & APAC bei Pentaho. Durch die Vernetzung von Maschinen und internetfähigen Alltagsgegenständen entstehen ständig neue Datenströme, die für Prognosen herangezogen werden können.

Neue Steuerungsmöglichkeiten: »Nur zu wissen, was passieren wird, bringt ein Unternehmen nicht weiter. Entscheidend ist, was mit dem vorhergesagten Szenario getan werden muss«, so SAS-Expertin Tschauder. »Erst die Kombination von Echtzeitanalysen und automatisierten Geschäftsprozessen bringt Anwendern den Effizienzgewinn, den sie im harten internationalen Wettbewerb benötigen«, erklärt Dunja Riehemann von Blue Yonder, einem Anbieter von Predictive-Analytics-Lösungen auf Software-as-a-Service-Basis.

PA-Methoden

Klassische statistische Verfahren kommen auch bei Predictive Analytics zum Einsatz. Dabei muss man zwischen linearen und nicht linearen Analysen beziehungsweise Vorhersagemodellen sowie parametrischen und nicht parametrischen Verfahren unterscheiden. Parametrische Verfahren gehen von einer Normalverteilung der Daten aus, der klassischen Gaußschen Glockenkurve. Parameterfreie Ansätze wie der Mann-Whitney-U-Test kommen dagegen ohne Annahmen über die Verteilung der Daten aus.

Zu den linearen Verfahren gehören die multiple lineare Regression (MLR), Partial Least Squares (PLS), Ridge Regression oder LDA (Linear Discriminant Analysis). Zu den nicht

Interview

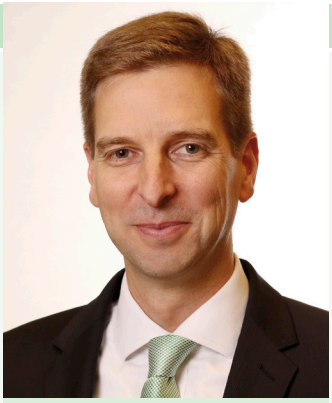
Neue Prognoseverfahren für Unternehmen

Carsten Bange erklärt im Gespräch mit **web & mobile developer**, was er unter fortgeschrittenen Analysen versteht und wo Unternehmen die neuen Prognoseverfahren einsetzen können.

web & mobile developer: *Lassen Sie uns zunächst die Begriffe klären: Was verstehen Sie unter Predictive Analytics? Wie unterscheiden Sie den Begriff zum Beispiel von Data Mining oder auch Prescriptive Analytics?*
Carsten Bange: Wir bei BARC bezeichnen diese Verfahren insgesamt als fortgeschrittene Analysen, um sie von einfachen Analysen, zum Beispiel durch Berechnung neuer Kennzahlen über die Grundrechenarten, abzugrenzen. Data-Mining-Verfahren haben darin eine beschreibende Funktion. Man nimmt die vorhandenen Daten und versucht, in ihnen Strukturen zu erkennen, etwa durch Segmentierung oder Assoziationsverfahren. Ein typisches Beispiel dafür ist die Warenkorbanalyse. Predictive Analytics im engeren Sinn sind vorhersagende Verfahren. Ich möchte eine Prognose treffen. Entweder möchte ich vorhersagen, in welche Klasse ein Objekt fällt. Typisches Beispiel ist die Risikoabschätzung bei der Kreditvergabe. Die zweite Variante ist eine Zeitpunkt- und Wertprognose. Ich sage etwa voraus, dass ich morgen in Filiale X 100 Stück von Produkt Y verkaufen werde.

»Predictive Analytics ist einer der Hauptgründe, weshalb Unternehmen Hadoop-Cluster aufbauen.«

Prescriptive Analytics schließlich ist ein recht neuer Begriff. Ähnliche Ansätze sind aber schon länger als Optimierungsverfahren oder Operations Research bekannt. Diese Verfahren berücksichtigen zusätzliche Umgebungsbedingungen wie Kosten oder Lieferfähigkeit und versuchen Entscheidungen so zu optimieren, dass beispielsweise der Umsatz maximiert und die Kosten minimiert werden. So kann es etwa besser sein, um bei unserem Beispiel zu bleiben, von den 100 verfügbaren Stück nur 90 von Produkt Y in Filiale X vorzuhalten, weil ich vielleicht in Filiale Z für die restlichen zehn einen besseren Preis erziele oder durch Zusatzgeschäfte insgesamt einen höheren Deckungsbeitrag erreiche.



Carsten Bange
Geschäftsführer des Analystenhauses BARC
www.barc.de

web & mobile developer: *Wie funktionieren diese Analysen?*
Bange: Generell versucht man entweder, Strukturen in Daten zu identifizieren, typischerweise Gruppen oder das zusammenhängende Auftreten von Attributen, oder man versucht durch sogenanntes überwachtes Lernen die Zusammenhänge zu finden, die zu einem bestimmten Ergebnis führen – in Zeitreihenanalysen oder der Vorhersage von Klassenzugehörigkeit. Hier kommen statistische Verfahren wie die Cluster-Analyse oder Regression zum Einsatz. Zu den spezielleren Methoden gehören neuronale Netze, und im Bereich Klassifikationsanalysen sind Entscheidungsbäume traditionell sehr beliebt. Für das Thema Assoziierung gibt es eigene Algorithmen.

web & mobile developer: *Welche Unternehmen profitieren am meisten von den neuen Methoden?*
Bange: Im Grunde können alle Unternehmen aller Branchen da-

linearen zählen LOWESS (Locally Weighted Scatterplot Smoothing), GAM (Generalized Additive Model), MARS (Multiple Adaptive Regression Splines) oder FDA (Flexible Discriminant Analysis).

Hinzu kommen analytische Prognosemodelle, die häufig auf maschinelles Lernen mit Hilfe neuronaler Netze setzen. Das System entdeckt eigenständig oder angeleitet Gesetzmäßigkeiten auf Basis von Trainingsdaten. Der daraus abgeleitete Algorithmus wird an Testdaten überprüft und optimiert. Nach einigen Durchgängen kann das Verfahren dann auf Daten angewandt werden, deren Klassifikation unbekannt ist. Zu den eingesetzten Verfahren gehören SVM (Support Vector Machine) und RVM (Relevance Vector Machine), Entscheidungsbäume (Decision Trees), Bootstrap Aggregation (Bagging) und Random Forests.

Die Prognosegüte steht und fällt mit der Datenbasis, weshalb dem Data Mining eine große Bedeutung zukommt. Alle internen verfügbaren Datenquellen zu entdecken, auszuwerten und zusammenzuführen sind vorbereitende Schritte in diesem Prozess. Menge allein genügt nicht, auch die Datenqualität muss stimmen. »Der typische Data Scientist verbringt 80 Prozent seiner Zeit damit, saubere Daten zu bekommen«, sagt Pentaho-Manager Claßen. Ein Data Warehouse ist zur Datenhaltung nicht zwingend notwendig. Es kann sogar kontraproduktiv sein, wenn die Daten zu sehr aggregiert und vorverarbeitet sind.

Einsatzszenarien

Bessere Prognosen kann praktisch jeder gebrauchen. Deshalb sind die Einsatzszenarien vielfältig. Microsoft-Manager Hans Wieser nennt IT-Sicherheitssysteme wie Intrusion Detection and Prevention sowie die Betrugsprävention bei Finanzdienstleistern und Telekommunikationsunternehmen als typische Anwendungsbeispiele. ▶

raus einen Nutzen ziehen. Ein Schwerpunkt liegt aber sicherlich im Handel. Dort profitiert man enorm von besseren Absatzprognosen und kann zum Beispiel die Sortimentsgestaltung, aber auch Lagerhaltung und Logistik besser steuern.
In produzierenden Unternehmen ist Predictive Maintenance das heißeste Thema, also die Möglichkeit, Wartung und Service von festen Zyklen auf ein datenbasiertes Modell umzustellen, das anhand der Maschinendaten die reale Abnutzung feststellen und bevorstehende Störungen prognostizieren kann.

web & mobile developer: *Welche Fachabteilungen setzen fortgeschrittene Verfahren vor allem ein?*
Bange: Vorreiter sind häufig die Marketing-Abteilungen. Sie nutzen Data Mining schon seit Längerem für die Kampagnenplanung. Wem schicke ich wann eine E-Mail? Wem mache ich welches Angebot? Als Zweites profitiert der Vertrieb von fortgeschrittenen Analysen, die beispielsweise bessere Absatzprognosen ermöglichen, aber auch der Einkauf – und hier schließt sich der Kreis: Wenn ich bessere Vorhersagen aus der Produktion oder dem Vertrieb bekomme, kann ich genauer einkaufen. Interessant für den Einkauf sind auch Handelsthemen, etwa die Vorhersage von Rohstoffpreisen an Börsen.

»Interessant für den Einkauf sind auch Handelsthemen, etwa die Vorhersage von Rohstoffpreisen an Börsen.«

web & mobile developer: *Nutzen Unternehmen dafür nur interne Daten oder greifen sie auch auf öffentlich verfügbare Informationen zu?*
Bange: Wir sehen generell den Trend, mehr externe Daten einzubeziehen, denn die Ergebnisgüte lässt sich dadurch häufig deutlich verbessern. Das Wetter hat zum Beispiel einen enormen Einfluss auf ganz viele Themen, weil es das Verhalten der Menschen stärker beeinflusst, als meist angenommen wird. Social-Media-Daten werden im Marketing gern analysiert, wobei der Nutzen eher im Consumer-Umfeld als im B2B-Bereich zu finden ist.

web & mobile developer: *Die Aggregation und Auswertung vieler Datenquellen erlaubt es, sehr genaue Profile von Kunden oder Anwendern zu erstellen. Ist der Datenschutz da gewährleistet?*
Bange: Das ist durchaus ein Problem, auch für diejenigen, die die Analysen durchführen. Unternehmen möchten sich ja in aller Regel rechtskonform verhalten, aber dennoch die Vorteile der fortgeschrittenen Analysemethoden nutzen. Da die rechtliche Situation teilweise unklar ist, ist das ein großes Problem, wenn es um personenbezogene Daten geht. Bei Maschinendaten ist die Lage natürlich deutlich entspannter.

web & mobile developer: *Muss ein Unternehmen in neue Hard- und Software investieren, um die Analysen nutzen zu können?*
Bange: Wahrscheinlich ja. Die klassischen Business-Intelligence-Lösungen decken die Anforderungen meist nicht ab. Wir sehen aber durchaus die Tendenz, dass die Anbieter zusätzliche Tools für fortgeschrittene Analysen mit ins Portfolio aufnehmen oder auch den Funktionsumfang bestehender BI-Werkzeuge erweitern.

web & mobile developer: *Benötige ich zwingend ein Hadoop-Cluster, um fortgeschrittenen Analysen betreiben zu können?*
Bange: Nein, man kann mit herkömmlichen relationalen Datenbanken viel erreichen, vor allem wenn man strukturierte Daten in nicht allzu großer Menge hat.
Hadoop ist immer dann das Mittel der Wahl, wenn es um semistrukturierte oder unstrukturierte Daten geht oder eben um riesige Menge von Daten, so ab einem zwei- bis dreistelligen Terabyte-Bereich.

Heleen Snelting von Tibco zählt die Betrugserkennung in Echtzeit, prädiktive Instandhaltung (Predictive Maintenance), die Routenoptimierung, das Supply-Chain- und Risikomanagement, das Kundenmarketing und das Mobile Asset Management zu den Haupteinsatzgebieten.

Eine Branche, die sehr stark von den neuen Vorhersagemethoden profitiert, ist der Handel. Amazon etwa generiert einen Großteil seines Umsatzes über die Produktempfehlungen, die jedem Kunden eingeblendet werden, der sich auf der Webseite des Online-Händlers eingeloggt hat. Sie beruhen auf fortgeschrittenen Analysen und beziehen eine Vielzahl von Faktoren wie das individuelle Kaufverhalten, den aktuellen Warenkorb, die lang- und kurzfristigen Abverkaufszahlen eines Produkts, die Preisentwicklung und sogar Erwähnungen in Tweets und Likes mit ein.

Im öffentlichen Raum sind es vor allem die Bereiche Verkehrssteuerung, Energie und Verbrechensprävention, auf denen die größten Hoffnungen liegen. »Wenn ich Verkehrsflüsse besser verstehen und Staus vermeiden will, wenn ich Grids (Stromnetze) besser managen will, um Blackouts zu vermeiden, wenn ich alternative Energiequellen intelligenter integrieren will, brauche ich prädiktive Modelle«, beschreibt Christian Dornacher, Director, Storage and Analytics Solutions EMEA bei Hitachi Data Systems, die Entwicklung der Städte zu »Smart Cities«.

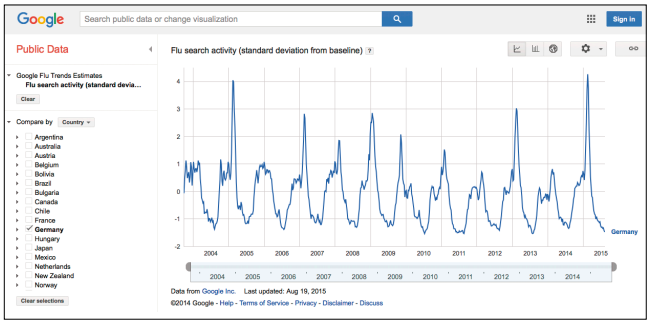
Zu erwähnen ist noch das Gesundheitswesen, das Predictive Analytics bei der Prognose von Krankheiten, aber auch in der Vorsorge und bei der Berechnung von Epidemieverläufen einsetzt. So hat beispielsweise Google mit Flu Trends eine Möglichkeit gefunden, Grippewellen vorherzusagen (Bild 1). Eine Analyse von Milliarden Datensätzen aus den vergangenen zehn Jahren ergab 144 Suchbegriffe, die für ein tagesgenaues Vorhersagemodell relevant sind.

Mit Predictive Analytics sind nicht nur erstaunlich genaue Vorhersage-, sondern auch völlig neue Geschäftsmodelle möglich. So setzt die in Deutschland umstrittene Taxi-Alternative Uber auf fortgeschrittene Analysemethoden. Der Erfolg des Unternehmens hängt unter anderem davon ab, dass es sehr genau vorhersagen kann, wo der größte Bedarf an Mitfahrgelegenheiten entstehen wird, ja sogar, wohin die Fahrgäste gebracht werden wollen.

Ein weiteres Beispiel sind die Finanztechnologie-Unternehmen (FinTech), deren Geschäftsmodell auf der Auswertung von Social-Media-Daten beruht. Mit deren Hilfe konnten sie Scoring-Modelle entwickeln, die es erlauben, die Kreditwürdigkeit von Menschen einzustufen, die nicht über eine Kredithistorie verfügen.

So gelingt der Einstieg

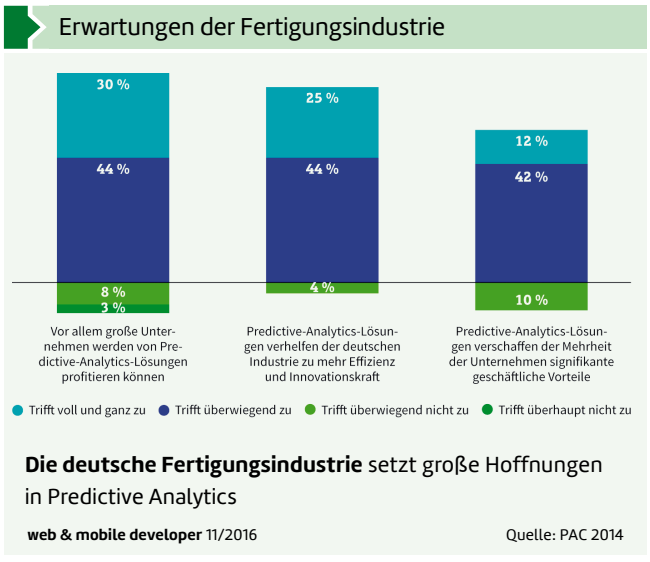
Datensammlung und Aufbereitung, die Wahl des richtigen Modells, Implementierung, Anpassung und schließlich die Interpretation der Ergebnisse stellen vor allem kleine und mittelständische Unternehmen vor erhebliche technische Herausforderungen. »Eine gute Zusammenarbeit zwischen den Fachabteilungen und der IT ist für die Erfassung der relevanten Daten und die Verwendung der richtigen Technologien notwendig«, sagt TIBCO-Managerin Snelting.



Predictive Analytics im Einsatz: Auf Basis von über 140 Suchbegriffen kann Google Grippewellen tagesgenau vorhersagen (Bild 1)

Krohn-Grimberghe von pmOne Analytics warnt allerdings vor einer rein technischen Betrachtung. Auch organisatorische und unternehmenskulturelle Aspekte spielten eine große Rolle: »Dazu zählt auch eine Führungspersönlichkeit mit dem Willen, das Unternehmen weiterzuentwickeln und analytische Themen voranzutreiben. Sonst verlaufen solche Initiativen oft im Sande.« Rado Kotorov, Chief Innovation Officer und Vice President Market Strategy bei Information Builders, einem Anbieter von BI- und Integrationslösungen, sieht dies ähnlich: »Die Herausforderung ist mehr kultureller als technischer Natur.« Analysen lohnten sich nur, wenn sie auch operationalisiert, das heißt in Anwendungen integriert würden, so Kotorov weiter. »Solange die Ergebnisse auf Papier oder in PowerPoint-Präsentationen bleiben und nicht in betriebsfähige Systeme aufgenommen werden, lässt sich kein Return on Investment erzielen.«

Auch Christian Dornacher von Hitachi Data Systems sieht die Herausforderungen weniger auf der technischen als auf der konzeptionellen Seite: »Man muss sich genau anschauen: Was will ich mit den Daten erreichen?« Dornacher empfiehlt, mit einem Workshop in das Thema einzusteigen, um diese Frage zu klären. Basierend auf den Resultaten dieses Workshops sollte das Unternehmen einen Lösungsansatz definieren und eine Roadmap erstellen, wie es die Ergebnisse



Die deutsche Fertigungsindustrie setzt große Hoffnungen in Predictive Analytics

web & mobile developer 11/2016 Quelle: PAC 2014

Tabelle 2: Werkzeuge für fortgeschrittene Analysen

Programm	Beschreibung	Internet
Apache Spark	Big-Data-Framework zur verteilten In-Memory-Berechnung auf Basis großer Datenmengen. Die Funktionsbibliothek MLib macht maschinelles Lernen für Spark-Systeme verfügbar.	https://spark.apache.org
Jupyter (IPython)	Interaktive Bedienoberfläche für über 40 Programmiersprachen zur Datenbereinigung und -transformation, für Simulationen, statistisches Modellieren, Machine Learning und andere Aufgaben.	https://jupyter.org
Python	Höhere Programmiersprache, die objektorientierte, aspektorientierte und funktionale Programmierung sowie eine dynamische Typisierung erlaubt. Python gilt als relativ einfach zu erlernen und wird meist in Form von Skripts angewendet.	www.python.org
R	Open-Source-Programmiersprache zur Modellierung statistischer Fragestellungen, die durch zahlreiche Pakete im Funktionsumfang erweitert werden kann. Im zentralen CRAN-Archiv (Comprehensive R Archive Network) finden sich über 5000 Pakete. R wird per Kommandozeile bedient, es existieren aber grafische Bedienoberflächen wie RStudio, R-Commander und relax. Außerdem gibt es kommerzielle Implementierungen wie Revolution R oder MapR.	www.r-project.org
Scala	Steht für Scalable Language; skalierbare, objektorientierte Programmiersprache, die unter anderem von Twitter, LinkedIn und Intel zur Datenanalyse eingesetzt wird.	www.scala-lang.org
Weka	Abkürzung für Waikato Environment for Knowledge Analysis; Weka wurde an der neuseeländischen Waikato-Universität entwickelt. Weka erlaubt die Analyse von Klassifikationen und Assoziationen mit Hilfe von maschinellem Lernen.	www.cs.waikato.ac.nz/ml/weka

erreichen will. Über Evaluierungs- und Pilotphasen ließe sich dann die richtige Lösung finden. »Das ist klassisches Business-Consulting«, sagt Dornacher.

Krohn-Grimberghe von pmOne empfiehlt, in ein Analytics-Labor zu investieren: »Bereits mit günstigen Open-Source-Technologien lässt sich verproben, ob Use Cases zutreffen und weiterverfolgt werden sollten.« Erst wenn ein Szenario in einem solchen Labor Erfolg verspreche, sollte man mit der klassischen Projektarbeit beginnen, so Krohn-Grimberghe.

Auswahlkriterien

Ist geklärt, wie Predictive Analytics im Unternehmen eingesetzt werden soll und welchen Nutzen man sich erwartet, geht es an die Wahl der richtigen Werkzeuge (Tabelle 2). Heleen Snelting von Tibco nennt einige Voraussetzungen dafür:

- **Benutzerfreundlichkeit:** In praktisch allen Branchen besteht ein Mangel an Fachkräften mit guten Analysekenntnissen. Daher ist es wichtig, für Predictive Analytics eine Plattform auszuwählen, die in der Handhabung für die Nutzer einfach ist.
- **Flexibler Datenzugriff und flexible Datentransformation:** Die Lösung sollte es Anwendern ermöglichen, Daten aus verschiedenen Quellen zu kombinieren, ohne SQL-Anweisungen schreiben zu müssen. Folgende Datenquellen sollten dabei einbezogen werden können: Excel, Cloud-Datenquellen, Webservices, Big Data Stores, relationale Datenbanken, Analytic Databases und OLAP-Systeme.
- **Erweiterte Funktionen:** Tools zum Erstellen von Datenbeziehungen, Prognosen und prädiktiven Modellen für Regressionen und Klassifikationen sowie Clustering sind ebenfalls wünschenswert. So lassen sich bei komplexen Daten schnell fortschrittliche Analysen durchführen.
- **Skalierbarkeit und Sicherheit:** Eine Predictive-Analytics-Lösung sollte skalierbar sein, da die Anzahl an Anwendern in Unternehmen zunimmt. Je mehr Nutzer und Datenquellen vernetzt beziehungsweise angebunden werden, desto wichtiger wird die sichere Verwaltung einer solchen Plattform. Die Lösung sollte darüber hinaus alle Funktionen bereithalten, die ein Unternehmen jetzt und in absehbarer Zeit benötigt.

Bei einem datengetriebenen Ansatz entwickeln sich die Anforderungen meistens mit der Zeit weiter. Das bedeutet, dass Unternehmen heute nicht alle notwendigen Anforderungen verstehen und voraussehen können. Sie sollten sich daher beraten lassen und verschiedene Anbieter und Lösungsangebote vergleichen.

Fazit

Moderne Vorhersagemethoden sind für alle Unternehmen von Vorteil, besonders dann, wenn sie direkt mit automatisierten Aktionen verknüpft werden können, wie dies in Bereichen wie Marketing Automation oder Predictive Maintenance bereits der Fall ist.

Das Schürfen, Klassifizieren und Auswerten vorhandener Daten kann – oft in Kombination mit externen Quellen – sogar ganz neue Geschäftsmodelle entstehen lassen. ■

Dr. Thomas Hafen
ist seit mehr als 15 Jahren als Redakteur und Journalist tätig, unter anderem für die IT-Fachzeitschriften NetworkWorld Germany und ChannelPartner sowie die Fotoseiten Seen.by und Digitalkamera.de.
http://thomas-hafen.de

SOFTWARE-ENTWICKLER

Must-have-Skills der IT-Profis

Foto: Shutterstock / Robuard

Die Zeiten sind vorbei, in denen Programmier-Jobs belächelt wurden, denn gut ausgebildete Fachkräfte sind die Hoffnungsträger unserer Wirtschaft.

Große wie kleine Unternehmen wie beispielsweise das Berliner IT-Unternehmen Neofonie mit seinen 180 Entwicklern ringen um IT-Experten aus aller Welt (Bild 1).

Der Dekra Arbeitsmarkt-Report 2016 ermittelte, dass Absolventen eines IT-Studiums die besten Jobchancen haben. Die vielen ausgeschriebenen Stellen deuten aber auch auf einen Engpass in der Branche hin. Grundvoraussetzung für einen IT-Beruf ist zumeist ein Studium mit dem Schwerpunkt Informatik, was aber auch Quereinsteigern nicht die Motivation nehmen sollte. Letztlich entscheidet das Paket aus Know-how, Erfahrung und Soft Skills. Ob Studium oder Quereinstieg – die folgenden Must-have-Skills verhelfen Software-Entwicklern, ihr Talent bestmöglich zu nutzen (Bild 2).

Kommunikation als A und O

»Kollege hat ein Problem, wie kann ich helfen? Wollen wir uns das gemeinsam anschauen? Zeig mal her!« – diese proaktive Vorgehensweise ist Alexandra-Irina Nicolae, Projektmanagerin von Neofonie, besonders wichtig. Gelerntes in der Praxis anzuwenden ist zwar das Ziel eines jeden Studiums, doch im beruflichen Umfeld stoßen Experten oftmals auf schier unlösbare Probleme. Informatiker sollten über die Stärke verfügen, nach Hilfe zu fragen. Hilfe anzubieten oder Unterstützung einzufordern sind wesentliche Qualitätsmerkmale eines Entwicklers.

Diese Stärke zeigt auch ein gesundes Selbstbewusstsein, das in der Branche verlangt wird. Der IT-Profi von heute sollte Ideen, Konzepte und Problemstellungen den Kollegen einfach erklären und zielführend über etwaige Lösungsansätze debattieren können, was auch in interdisziplinären Teams funktionieren muss.

Karsten Rieger, Senior Frontend Developer von Neofonie ist sich sicher: »Aus meiner Sicht gewinnen interdisziplinäre Fähigkeiten immer mehr an Bedeutung. Ein Team, das aus hochgradigen Spezialisten besteht, die tief in ihren Domänen verwurzelt sind, aber nicht über ihre Bereichsgrenzen blicken wollen oder können, wird nie so effizient arbeiten können wie ein Team, das neben Spezialisten auch solche Mitglieder aufweist, die an den Schnittstellen mehrerer Domänen arbeiten können.«

IT-Experten sind in IT-Unternehmen wie der Berliner Neofonie mit seinen 180 Entwicklern gefragt (Bild 1)

Lösungsorientiertes Denken

Schnell entwickeln sich Softwareprojekte zum Spießrutenlauf. Der Kunde reagiert während der Projektphase auf Veränderungen im Markt und fordert mitunter aufwendige Anpassungen in der Software. In solchen Situationen abzulehnen ist schlicht der falsche Weg. Der IT-Spezialist muss sich in den Kunden hineinversetzen und Alternativen aufzeigen können, wenn es darum geht, ein Softwareprodukt zu verbessern oder eventuelle Probleme zu be-

heben. Malte Cornils, Senior System Architect von Neofonie, gibt zu bedenken, dass Entwickler im Kundenkontakt sich die Fragen »Wie erkläre ich dem Kunden das Lösungskonzept?« und »Welche Vor- und Nachteile sind damit verbunden?« beantworten und diese Antwort auch geschickt formulieren können müssen. Realsituationen bei Neofonie: Direkter Kundenkontakt, Supportanfragen, Planungsmeetings, Konzeptphase, Sales, Pitch Präsentation, Entwicklungsphase.

Wie in allen Bereichen ist Zuverlässigkeit unabdingbar. Terminlich und qualitativ müssen Deadlines eingehalten werden, um das Erreichen des Endprodukts nicht zu gefährden. Gegen ablaufende Fristen haben Entwickler-Teams als auch Auftraggeber gleichermaßen zu kämpfen. Um diese Bredouille zu vermeiden, sind gutes Teamwork und eine solide Kommunikationsbasis notwendig, die zahlreiche Projektmanagement- und Kollaborations-Tools bieten.

Teamplayer statt Einzelkämpfer

Als geschlossenes Team aufzutreten ist wichtig. Fehler einzugestehen und Selbstkritik zu äußern sind keine Zeichen von Schwäche. Im Gegenteil – Fähigkeiten selbstkritisch zu betrachten und sich neue Verhaltensmuster anzueignen ist im IT-Sektor gefragter denn je. Es gibt nicht mehr einen Experten, es gibt unzählige Experten auch innerhalb eines Unternehmens. Da zählen der Zusammenhalt und die gegenseitige Unterstützung – und nicht der Einzelkämpfer.

Zusätzlich hat die Clean-Code-Developer-Initiative Tugenden aufgestellt, die ein teamorientierter Software-Entwickler auf alle Fälle an den Tag legen sollte. Teamplay fängt bei der Code-Semantik an, da ist sich auch Christian Springsguth, System Architect von Neofonie, sicher: »Wichtig für einen Entwickler ist, dass er mit dem Code eine Absicht ausdrücken kann. Der Code muss verraten, warum etwas getan wird. Das ist meist wichtiger als zu verraten, wie etwas getan wird. Das ist eine Voraussetzung dafür, dass der Code auch von anderen verstanden und weiterentwickelt werden kann.«

Frameworks, Programmiersprachen und CMS

Der griechische Philosoph Heraklit sagte einst: »Nichts ist so beständig wie der Wandel« – das trifft auch für die Software-Branche zu. Wer in der Programmier-Elite mitspielen möchte, muss sich ständig an den Marktanforderungen orientieren und sich weiterbilden, seien es Programmiersprachen, Frameworks, Webtechnologien oder Shop-Systeme. Auch Malte Cornils, Senior System Architect, bestätigt das: »Offenheit zum Lernen ist das eine Must-have, das einzelne Technologiekenntnisse schlägt. Allerdings sind organisatorische und technische Erfahrungen dadurch natürlich mindestens genauso wichtig.«

Karsten Rieger plädiert für den Mut zur Lücke. Es sei nicht nötig, alles bis ins Detail zu wissen, sondern viel mehr benötigt man Offenheit sowie Neugier, um andere Disziplinen kennenzulernen. Der Willen zur Kommunikation und nicht zuletzt auch eine Prise Humor schaden nicht, um sich selbst in der Hektik des Alltags nicht zu ernst zu nehmen. Wenn man es dann noch schafft, mit einem Auge auf das Gesamtbild der Produktanforderungen zu schauen, gelingt es am besten, den Überblick zu behalten. In diesem Fall wäre man seiner Meinung nach ebenfalls ein Top-Kandidat für eine Beschäftigung als Entwickler. Dementsprechend spielt auch beim Erlernen der Team-Gedanke eine wesentliche Rolle.

Absolventen eines IT-Studiums haben laut Dekra die besten Jobchancen (Bild 2)

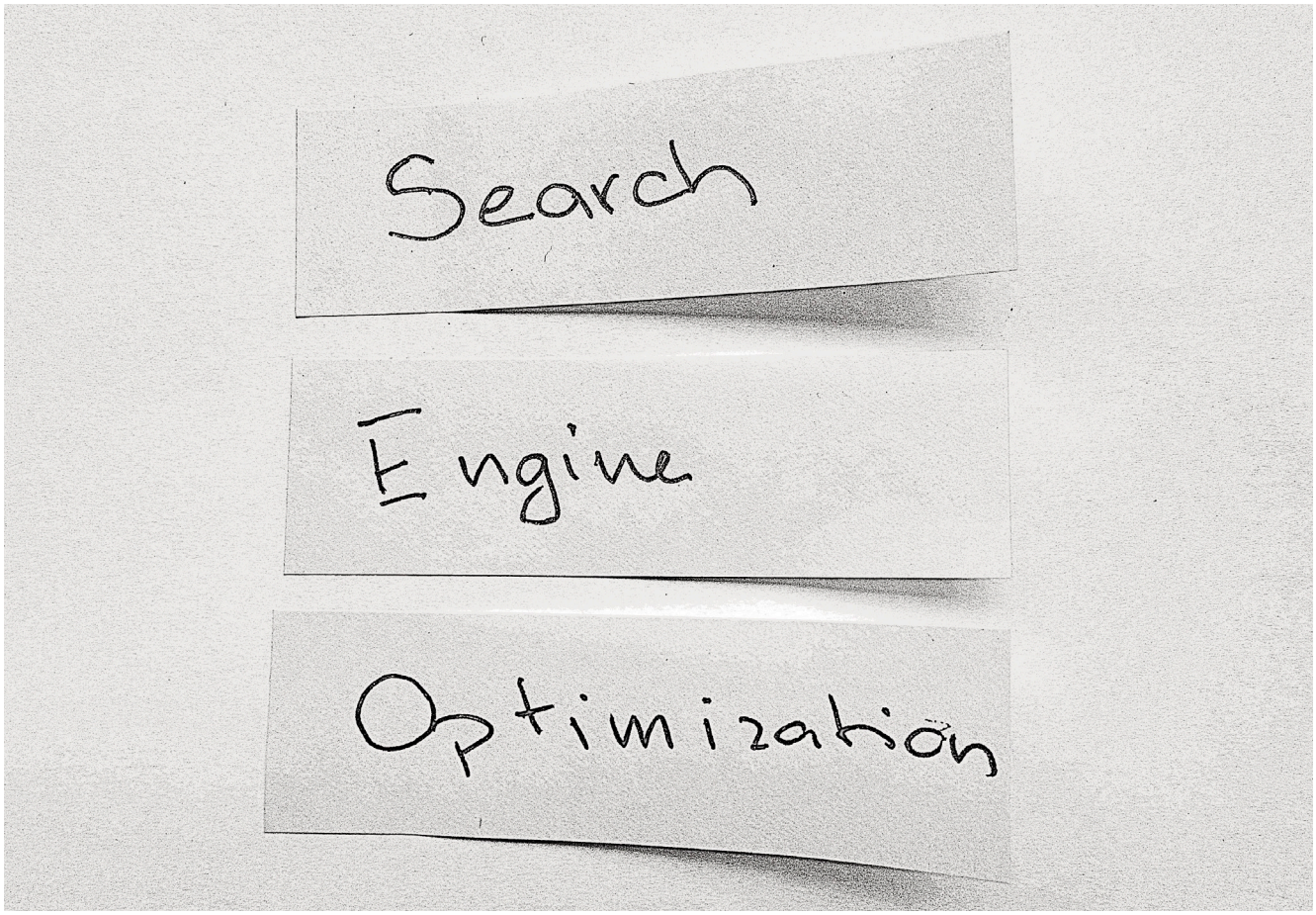
Einsatzbereitschaft trifft Flexibilität

Unvorhergesehene Ereignisse sind alles andere als selten. Inhaltlich fehlerhafte Absprachen oder unterschiedliche Interpretationen kennen Softwareteams nur zu gut. Besonders am Ende eines Projekts ist die Nervosität beim Kunden am höchsten und es fallen aufgrund kurzfristiger Änderungswünsche Überstunden an. In dieser Hinsicht spielt die Qualitätssicherung eine große Rolle, bei der Entwicklerkollegen unterstützend Einsatzbereitschaft und Flexibilität entgegenbringen sollten.

Fazit

Ob Start-up oder Konzern – Fachkräfte im IT-Bereich sind gefragt wie nie zuvor. Qualifizierte Informatiker müssen mittlerweile neben Enthusiasmus und Leidenschaft auch eine Reihe an Soft Skills mitbringen. Das Gesamtpaket aus sozialer Kompetenz und technologischen Know-how muss schlichtweg stimmen. Teamplayer statt Alleskönner, so lautet die Devise, denn IT-Experten mit dem gewissen Etwas gehört die Zukunft. ■

Ender Özgür
wurde 1972 in Berlin geboren und hat Informatik an der TU Berlin studiert. Er ist bereits seit 2001 Teil der Neofonie und verantwortet als Head of Software Factories die zeitgemäße Umsetzung von Entwicklungsprojekten.
www.neofonie.de



www.i-ways.net

SEO IM EINZELHANDEL

Schneller gefunden werden

Wie Händler ihre Online-Präsenzen optimieren können.

Suchmaschinenoptimierung gehört zu den wesentlichen Aufgaben, wenn Einzelhändler an der Digitalisierung ihres Geschäftsmodells arbeiten. Sie sollte stets zielbezogen sein – ein Ranking ist kein Selbstzweck. Außerdem gilt es, die Wechselwirkungen zwischen Social-Media-Aktivitäten und deren Bewertung durch Suchmaschinen zu nutzen. Digitalagenturen mit Expertenwissen und ganzheitlicher Herangehensweise können Händler dabei nachhaltig unterstützen.

Der Handel ist im Umbruch – Kunden erwarten zunehmend, zwischen mehreren Verkaufskanälen oder Touchpoints frei wählen und wechseln zu können. Diesen Erwartungen sollen und wollen Händler entgegenkommen, indem sie ihre Vertriebs- und Marketingkanäle um Internetplattformen und Social-Media-Präsenzen erweitern.

Ziel sollte es sein, eine Multichannel-Strategie zu entwickeln, die alle Kanäle – sowohl online als auch offline – einbezieht und somit das gesamte Potenzial zur Erreichung der Ziele ausschöpft. Die neu ausgebaute Online-Präsenz kommt jedoch nur dann zum Tragen, wenn man als Händler auf den entscheidenden Plattformen auch gefunden wird. Im Rahmen einer gesamten Marketingstrategie spielt deshalb Suchmaschinenoptimierung eine wichtige Rolle.

Stationäre Händler mit oft jahrelanger Erfahrung sehen sich dabei vor der Aufgabe, Bekanntheit und Reputation, die sie sich bei ihren Kunden im Ladengeschäft über die Jahre erarbeitet haben, auf den Online-Markt zu übertragen. Dabei müssen die Algorithmen von Suchmaschinen und Marktplätzen verstanden und durch kontinuierliche, auf den Nutzer ausgerichtete Maßnahmen genutzt werden, um ein stringentes Markenbild aufzubauen und gute Rankings zu erzielen.

SEO ist nicht losgelöst zu betrachten

Suchmaschinenoptimierung (Search Engine Optimization, SEO) wird stark beeinflusst von Nutzersignalen und ist deshalb stets im Kontext von Social-Media-Aktionen zu sehen. Richtig eingesetzt bieten Social-Media-Kanäle die Chance, die Interaktion mit den Kunden auf einer weiteren Ebene positiv zu gestalten und die Kundenbindung zu stärken. Emp-

fehlungsmarketing wirkt sich positiv auf die Suchergebnisse aus. Dafür bedarf es einer übergreifenden Strategie, kontinuierlicher Pflege und Monitoring.

Ein Social-Media-Profil zu erstellen und dann nur einige Beiträge zu posten verschwendet nicht nur wertvolle Zeit, sondern birgt sogar Risiken: Wer Fragen der Nutzer auf seiner Facebook-Seite nicht beantwortet, riskiert verärgerte Kunden und leider – der niedrigeren Hemmschwelle auf diesen Plattformen geschuldet – oft drastische Kommentare.

Wird umgekehrt eine Information zu Rabattaktionen im Ladengeschäft nur über Facebook gepostet, nicht aber auf der Website und im Online-Shop, führt das zu Verdruss bei den nicht informierten Online-Käufern, was wiederum zu negativen Kommentaren und Bewertungen führen kann.

Suchmaschinen wie Google erfassen in ihren Algorithmen auch solche Interaktionen als Social Signals. Somit haben auch Emotionen – positive wie negative –, die mit einer Marke verknüpft sind, eine Auswirkung auf das Ranking (Bild 1).

Ein anderes Beispiel: Gute Bewertungen eines Unternehmens bei Google My Business führen zu höheren Klickzahlen und resultieren im besten Fall auch im Besuch des Ladengeschäfts und einer Weiterempfehlung.

Weist aber die lokale Suche im Netz eine veraltete Adresse oder falsche Öffnungszeiten aus, wird das ein enttäuschter Kunde, der den Weg umsonst gemacht hat, wahrscheinlich mit einer entsprechenden Bewertung bei Google My Business quittieren.

Diese Negativ-Bewertung erscheint danach ebenfalls in der Trefferanzeige. Andere Kunden werden die Website daraufhin wahrscheinlich seltener anklicken, was wiederum vom Google-Algorithmus erfasst wird – im Ergebnis fällt die Seite im Ranking ab (Bild 2).

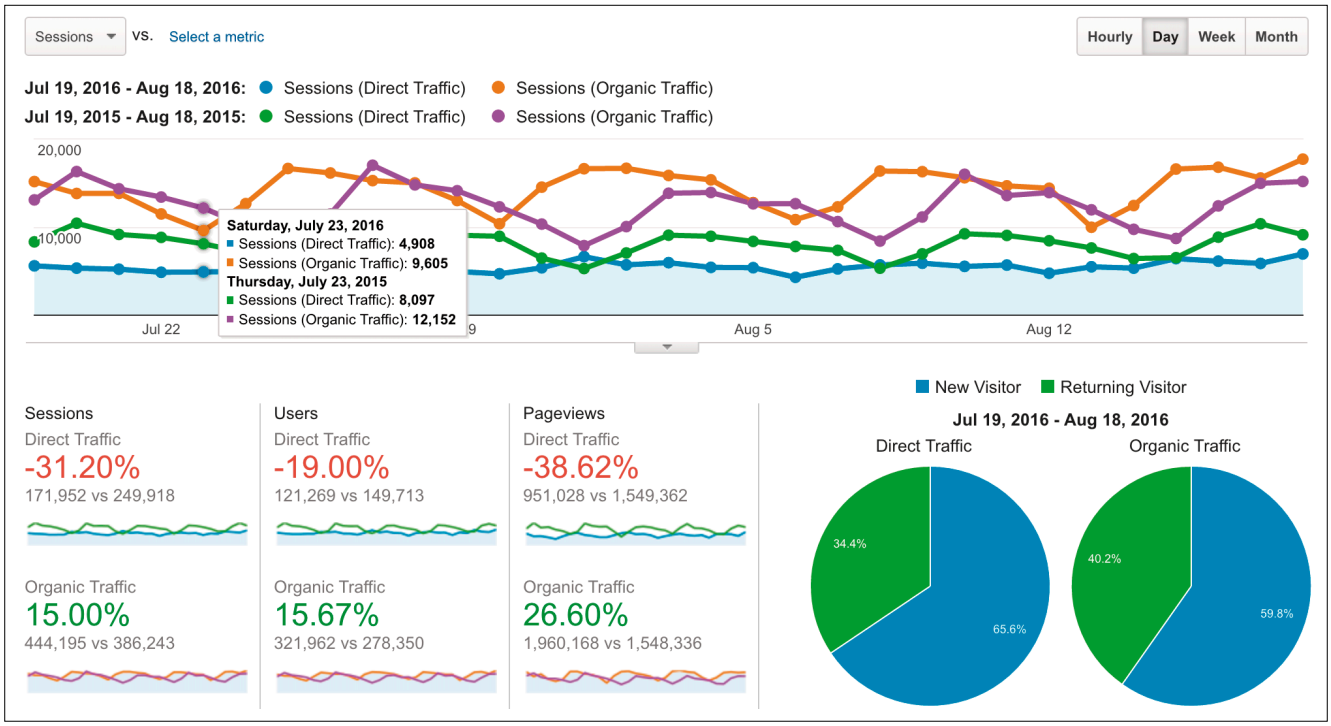
Wie sollte man die Aufgabe also am besten angehen, um online besser gefunden zu werden? Wenn es an eigenen Ressourcen im Online-Marketing mangelt, kann man auch auf die Hilfe externer Dienstleister und spezialisierter Digitalagenturen zurückgreifen.

Diese gehen systematisch vor. Die erste Frage dabei ist immer: Was soll mit einer Optimierung erreicht werden? Es geht nämlich gar nicht einfach um ein besseres Ranking. Maßgeblich sind immer die angestrebten Ziele, für deren Erreichung ein besseres Ranking die Voraussetzung ist. Diese können sein: Absatzsteigerung, eine deutlichere Markenpräsenz oder auch mehr Kunden auf der Fläche. Entsprechend sind die Plattformen auszuwählen, auf denen der Händler aktiv wird. Es gibt keine Strategie, die für jedes Unternehmen passt. Für Instagram sind Produkte geeignet, die sich gut inszenieren oder in ein Ambiente setzen lassen. Wer absehbar kaum twittertaugliche Nachrichten zu verkünden hat, benötigt auch keinen Twitter-Account.

Vorteile stationärer Händler

Stationäre Händler haben gegenüber reinen Online-Shops den großen Vorteil, in ihren Ladengeschäften Kunden im persönlichen Gespräch beraten zu können. Click and Collect, Reservierungen im Ladengeschäft und virtuelle Regalerweiterung bieten weitere Möglichkeiten, sich von Online-Shops abzuheben. Deshalb ist die Stärkung der Ladengeschäfte sinnvoll, und lokale SEO-Maßnahmen spielen eine entscheidende Rolle.

Die Optimierung wird dabei auf die kombinierte Suche aus Begriffen zum Produkt und generischen Suchbegriffen ausgerichtet. Am Anfang steht eine Bedarfserhebung, eine Analyse des technischen Zustands der Website und der Social- ▶



Suchmaschinenoptimierung (SEO) wird stark beeinflusst von Nutzersignalen (Bild 1)

Media-Profile mit Bezug auf den Wettbewerb. Dabei verlässt man sich nicht auf die gefühlte Konkurrenz, sondern recherchiert zunächst relevante und kontextbezogene Keywords. Unterschieden wird zum Beispiel nach Informationskontext, wie »Bewertung Produkt X«, und Transaktionskontext, wie »Produkt X kaufen«, für Local SEO noch erweitert um entsprechende Keywords zur lokalen Suche. Die Technologien zur Keyword-Recherche sind abhängig von den im jeweiligen Land dominierenden Suchmaschinen. Anhand der definierten Keywords ermittelt die Agentur die Top-Positionen der Google SERP (Search Engine Result Page). Die bestplatzierten Mitbewerber des Nischenbereichs sind die, mit denen der Kunde verglichen wird. Deren Aktivitäten werden analysiert, um zu sehen: Welche Maßnahmen sind in dieser Branche üblich, welches sind die typischen Plattformen? Aber auch: Was kann der Kunde besser, wie kann er sich mit seinem spezifischen Angebot vom Wettbewerb absetzen?

Gute Bewertungen bei Google My Business führen zu höheren Klickzahlen (Bild 2)

On-Page-Optimierung

Auf dieser Basis wird optimiert – sowohl nach grundlegenden Anforderungen der On-Page-Optimierung, wie relevantem und übersichtlich strukturiertem Content, als auch nach den entsprechenden Kriterien der einzelnen Plattformen. Es ist wichtig, den Suchgedanken der jeweiligen Suchmaschine, die Nutzerintention, zu verstehen. So sind bei Amazon sehr schnelle Lieferzeiten und Reaktionen auf Beschwerden ein wichtiges Ranking-Kriterium. Bei Ebay steht eher der Preis im Vordergrund. Dementsprechend müssen auch Texte und Bilder passend aufbereitet werden.

Für die Bekanntmachung von Blogs sind Gastbeiträge oft das Mittel der Wahl – entsprechende Kontakte zu wichtigen Influencern können ebenfalls Digitalagenturen vermitteln. Hierbei nutzt man den Nachahm-Effekt: Sehen Nutzer, dass auf einer Seite bereits Interaktionen stattfinden, sind sie eher geneigt, auch selbst zu reagieren. Häufigere Besuche, Kommentare und Erwähnungen wiederum sorgen dafür, dass neue Inhalte schneller von den Suchmaschinen übernommen und angezeigt werden, im besten Fall schneller als die der Mitbewerber. Mit Aktionen wie Gutscheinen für einen Wiedereinkauf lassen sich auch dauerhaft Kunden gewinnen.

Nachhaltige Lösung sichert den Erfolg

Wie auch im letzten Beispiel deutlich wird, ist es mit einer einmaligen Optimierung nicht getan. Pflege, Monitoring und ständige Anpassung der Seiteninhalte und Social-Media-Präsenzen bedeuten dauerhaften Aufwand und fordern den Einsatz von Ressourcen. Bei der i-ways sales solutions GmbH widmet man diesem erfolgskritischen Aspekt besondere Aufmerksamkeit: Projektbegleitend werden die Mitarbeiter im Verkauf durch ein Trainerteam geschult.

Im Ergebnis könnte dann beispielsweise eine Anfrage über Facebook zu einem bestimmten Produkt vor Ort direkt an einen Mitarbeiter der nächsten Filiale weitergeleitet und von diesem persönlich beantwortet werden. Zusätzlich kann auch eine Schlüsselperson in der Marketingabteilung für die Aufgaben als Online-Manager durch die Agentur geschult werden. Dieser wird dann als Experte im Unternehmen sein Wissen weitergeben und gleichzeitig als zentraler Kontakt die Verbindung von Marktwissen des Händlers mit der Online-Expertise der Digitalagentur sicherstellen. Bei geringeren personellen Ressourcen ist die Aufgabenteilung zwischen Agentur und Marketing des Händlers im Rahmen einer langfristigen Zusammenarbeit gängige Praxis, etwa indem ein Blog intern betreut wird, während die Agentur die Pflege der Social-Media-Plattformen übernimmt. Selbst die Managementkontrolle für alle Plattformen des Kunden, also auch die Prüfung von Pressemeldungen und anderen Veröffentlichungen auf SEO-Eignung, lässt sich auslagern.

Fazit

Alle Maßnahmen zur Suchmaschinenoptimierung und Pflege der Social-Media-Präsenzen sollten von den Geschäftszielen abgeleitet und unter klarem Kosten-Nutzen-Aspekt betrachtet werden. Überall dort, wo es dem Händler nützt, kommt es darauf an, präsent und gut aufgestellt zu sein – mit den richtigen Botschaften auf den richtigen Kanälen. ■

Florian Laudahn
ist CMO bei der i-ways sales solutions GmbH, einer Digitalagentur mit 70 Mitarbeitern und Niederlassungen in sieben Ländern.
www.i-ways.net

ITUNES CONNECT

Neuerungen und Änderungen

Auf der WWDC bot Apple einen Ausblick auf neue Features von iTunes Connect.

Für alle Apple-Entwickler ist iTunes Connect die Plattform, um die eigenen Apps im App Store zu veröffentlichen, Verkaufsreports einzusehen, statistische Informationen zu erhalten und auszuwerten, Werbecodes zu generieren und herunterzuladen und vieles mehr. Kurzum ist es die Plattform, die nach der eigentlichen Entwicklung einer App alle darauf folgenden und benötigten Funktionen vereint (Bild 1).

iTunes Connect wurde dabei bereits in den letzten Jahren von Apple stetig gepflegt und weiterentwickelt und hat inzwischen sogar einen kompletten Relaunch hinter sich. Auf der diesjährigen WWDC sprach Apple dann in einer eigens dafür bereitgestellten Session über die kommenden Neuerungen und Änderungen, die iTunes Connect bis Ende dieses Jahres erfahren soll. Einige von den in der Session aufgeführten Punkten sind dabei inzwischen bereits umgesetzt, andere folgen erst noch in den nächsten Wochen und Monaten. In diesem Beitrag werden Ihnen alle genannten Neuerungen einmal im Detail vorgestellt.

Subscriptions für alle App-Kategorien

Mit Hilfe von Subscriptions ist es möglich, eine Art Abo-Modell für eigene Apps umzusetzen, durch die ein Nutzer regelmäßig für eine App bezahlt. Magazin-Apps können darüber beispielsweise Abonnements direkt über die App-Store-Infrastruktur umsetzen.

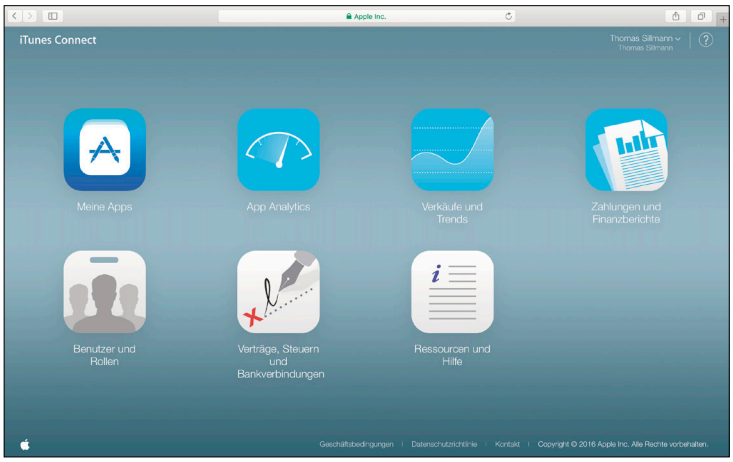
Diese Technik öffnet Apple nun. Bisher waren Subscriptions nur für einzelne, von Apple vorgeschriebene App-Kategorien verfügbar, nun soll es mit allen Kategorien möglich werden, Subscriptions zu erstellen. In diesem Zuge führt Apple auch das sogenannte Territory Pricing ein. Damit lassen sich für verschiedene Zielgebiete unterschiedliche Price Tiers einer App festlegen, statt wie bisher einen Price Tier, der überall und in allen Regionen gleich ist. Dabei wird es gleichzeitig auch neue Price Tiers geben, um die Preise von Apps und In-App-Käufen noch besser anzupassen und freingranularer festzulegen.

Ein neues Feature namens Price Preservation betrifft ebenfalls Subscriptions. Damit können Sie festlegen, dass bei einer möglichen Erhöhung des Preises für eine Subscription bereits bestehende Abonnenten nicht den neuen, erhöhten Preis zahlen müssen, sondern stattdessen den vergünstigten behalten, der bei Abschluss ihres Abonnements galt. Sie werden erst dann auch den höheren Preis zahlen müssen, wenn sie ihr Abonnement einmal beenden und erneut abschließen.

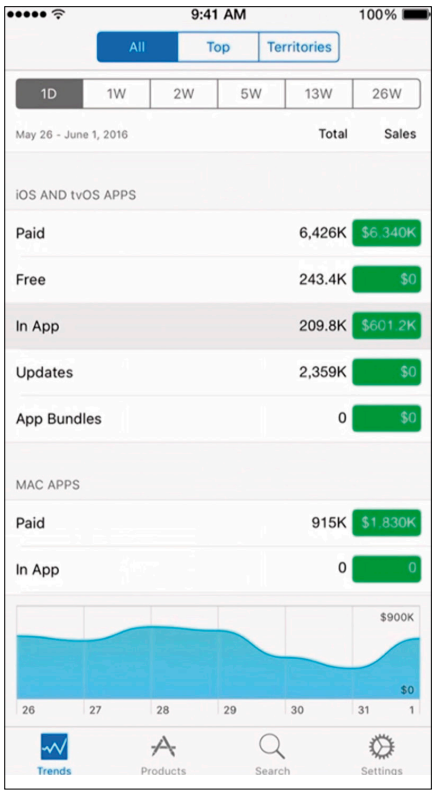
Zu guter Letzt ändert Apple auch ein wenig die Preispolitik im Bereich Subscriptions: Abonnenten, die mindestens ein Jahr ein Abonnement bei einer App abgeschlossen haben, bringen dem zugehörigen App-Entwickler ab dann 85 Prozent der Erlöse und nicht – wie bisher – 70 Prozent. Damit beträgt die an Apple zu entrichtende Gebühr nur noch 15 Prozent statt wie bisher 30 Prozent. Diese Erlösstaffelung soll bereits jetzt gelten und bezieht existierende Subscriptions aus der Vergangenheit mit ein. Haben Sie also eine App im Angebot, die bereits seit über einem Jahr verfügbar ist, und verfügen Sie entsprechend über Abonnenten, die schon seit über einem Jahr diese Subscription abgeschlossen haben, profitieren Sie unmittelbar von den erhöhten Erlösen.

Update der iTunes-Connect-App

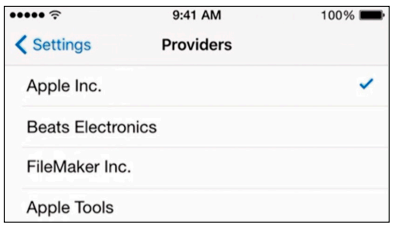
Eine der wahrscheinlich unter Apple-Entwicklern meistgenutzten Apps auf iPhone und iPad dürfte iTunes Connect sein. Über diese App ist es möglich, Verkaufszahlen und Bewertungen zu den eigenen Apps einzusehen sowie den Review-Prozess einer Binary zu verfolgen, die man zur Prüfung bei Apple für eine App eingereicht hat. Im Zuge der Neuerungen in iTunes Connect war es nur konsequent, auch der iTunes-Connect-App für iOS ein Upate zu verpassen. Bereits verfügbar ist der Zugriff auf die Gewinne und Erlöse, die bisher



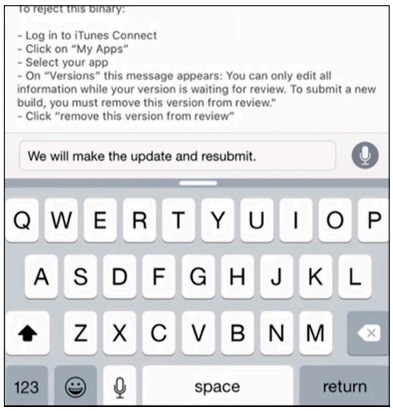
iTunes Connect bietet verschiedenste Tools für den Vertrieb der eigenen Apps und das Auswerten von App-Downloads (Bild 1)



Mittels Multiprovider-Support können App-Entwickler, die mehreren Teams angehören, künftig jedes Team aus der iTunes-Connect-App heraus wechseln (Bild 3)



Erzielte Erlöse und Gewinne lassen sich bereits jetzt direkt aus der iTunes-Connect-App heraus auslesen (Bild 2)



Die iTunes-Connect-App wird den direkten Zugriff auf das Resolution-Center inklusive Antwortmöglichkeit bieten (Bild 4)

glied von mindestens zwei verschiedenen Teams sind und somit die Informationen zu Apps aus unterschiedlichen Quellen einsehen möchten. In Zukunft wird es über die Einstellungen der iTunes-Connect-App möglich sein, aus einer vorgefertigten Liste aller Provider, denen die eigene Apple-ID zugeordnet ist, direkt zu wechseln und somit schnell und bequem die jeweils benötigten Informationen aus dem passenden Team abzurufen (Bild 3).

Auch das Resolution Center wird optimiert. Hat Apple beim Review-Prozess einer App etwas zu beanstanden, so findet sich in iTunes Connect zukünftig direkt der Verweis auf die zugehörige Passage aus den Review Guidelines von Apple. Darüber hinaus wird es möglich sein, direkt auf diese Rückmeldung von Apple aus der App heraus zu reagieren (Bild 4).

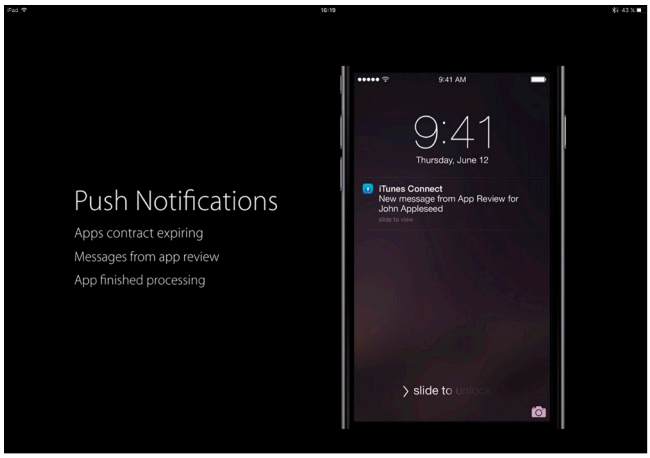
Zu guter Letzt werden die Notifications der App optimiert. Es werden neue dazukommen, die beispielsweise über eine ablaufende Mitgliedschaft des Apple Developer Program informieren, Nachrichten aus dem Review-Prozess von Apps beinhalten und die Bescheid geben, sobald eine neu eingereichte App oder ein App-Update das Processing für iTunes Connect abgeschlossen haben (Bild 5). Insbesondere Letzteres dürfte viele App-Entwickler freuen, kann das Processing doch einige Zeit in Anspruch nehmen und ein regelmäßiges Prüfen des Processing-Status nervig sein.

nicht über die App einsehbar waren. In der Übersicht der Verkaufszahlen kann aber nun per Tap auf die einzelnen Zellen zwischen den verschiedenen Informationen gewechselt und dabei nun auch auf die erzielten Gewinne und Erlöse direkt aus der App heraus zugegriffen werden (Bild 2).

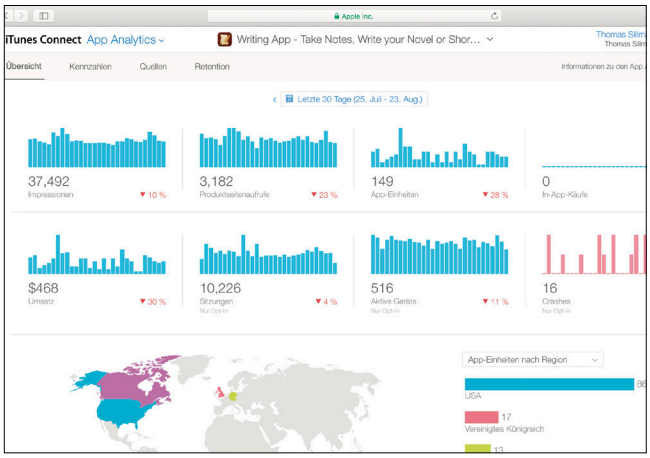
Darüber hinaus hat Apple für die Zukunft weitere Verbesserungen und Optimierungen für die App versprochen. Dazu gehören unter anderem der sogenannte Multiprovider-Support. Der ist für all jene App-Entwickler interessant, die Mit-

Erweiterung von App Analytics

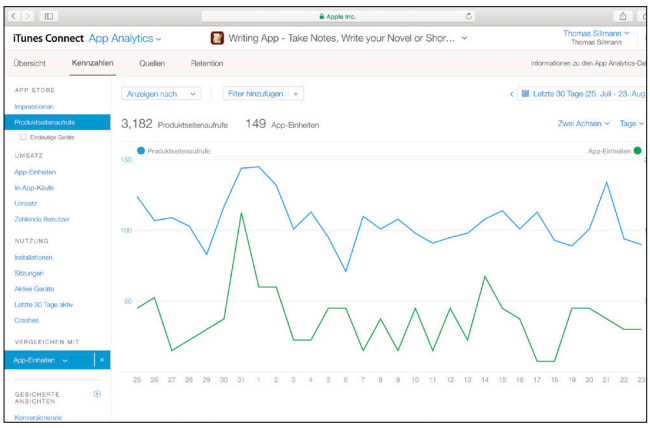
Mit App Analytics hat Apple bereits vor einiger Zeit einen eigenen Service eingeführt, der App-Entwicklern dabei helfen soll, das Potenzial ihrer App zu verbessern und mögliche Schwachpunkte im Marketing auszumachen (Bild 6). So er- ▶



Neue Notifications liefern regelmäßig Informationen über den Prozess einer App in iTunes Connect (Bild 5)



App Analytics liefert eine Vielzahl von Kennzahlen zu den eigenen Apps (Bild 6)



Informationen wie die Conversion Rate können nun mittels App Analytics ausgewertet werden (Bild 7)

fasst App Analytics die Anzahl der Aufrufe einer App-Store-Seite und kann diese in direkten Kontext zu den abgeschlossenen Käufen und Downloads setzen, um darüber die entsprechende Conversion Rate zu ermitteln (Bild 7). Auch wird App Analytics in Zukunft darüber informieren, aus welcher Quelle ein Nutzer die eigene App gefunden hat. Dabei unterscheidet der Dienst zwischen den Kategorien *App Store, Websites* und *Apps*.

Dabei fasst *App Store* alle Käufe und Downloads zusammen, die direkt aus dem App Store erfolgten, ganz gleich ob der Nutzer die eigene App über die Suche, die Charts oder die von Apple redaktionell gepflegten Listen gefunden hat. *Websites* verweist entsprechend auf Internetseiten, über die man zur eigenen App im App Store weitergeleitet wurde, während unter *Apps* all jene Apps aufgeführt werden, aus denen heraus die eigene App im App Store aufgerufen wurde.

iMessage und Promo-Codes

Auch die eigentliche Website von iTunes Connect optimiert Apple in mehrerlei Hinsicht. Neben den eben vorgestellten Verbesserungen bei App Analytics wird es in Zukunft möglich sein, Screenshots für eine App auf Wunsch nur noch einmalig pro Plattform und optional auch nur für eine einzige Sprache anzubieten. Damit entfällt das aufwendige Erstellen von Screenshots für die eigene App pro Gerätegröße und übersetzte Sprache (was aber optional natürlich weiterhin noch möglich ist). Für viele Apps dürfte das ein willkommenes neues Feature sein, sind die meisten Screenshots pro Plattform für den App Store doch identisch und unterscheiden sich lediglich minimal in Bezug auf deren Größe.

Ebenfalls wird im Zuge der mit iOS 10 neu eingeführten iMessages Extension auch eine neue Kategorie für Apps verfügbar gemacht, um so beispielsweise Sticker-Apps für die Nachrichten-App deutlich zu kennzeichnen. Auch zusätzliche Screenshots speziell für iMessages-Extensions werden in Zukunft möglich sein.

Zu guter Letzt wird Apple den Funktionsbereich seiner Promotion-Codes erweitern. Diese Promo-Codes können für eine App in iTunes Connect erzeugt werden und erlauben den kostenlosen Download dieser App. Dieser Mechanismus wird beispielsweise gerne dazu genutzt, die eigene App Bloggern oder anderen Webplattformen kostenfrei für einen Test bereitzustellen. Zukünftig wird man nun nicht nur in der Lage sein, Promo-Codes für die eigenen Apps, sondern auch für In-App-Käufe zu generieren.

Apple ist bemüht, die eigenen Guidelines zur Entwicklung von Apps für iOS und macOS in Zukunft weiter zu optimieren und auch leichter verständlich zu machen. Zu diesem Zweck wird es unter anderem in Zukunft fünf verschiedene Kategorien geben, in denen Apple passende Inhalte aus seinen Guidelines unterbringt: Safety, Performance, Business, Design und Legal. Darüber hinaus sollen die Guidelines in verschiedene Sprachen übersetzt und potenziell doppelte Inhalte, die sowohl iOS als auch macOS betreffen, vereinheitlicht und zusammengefasst werden.

Links zum Thema

- iTunes Connect
  https://itunesconnect.apple.com
- WWDC 2016-Session »What's New in iTunes Connect«
  https://developer.apple.com/videos/play/wwdc2016/305

Fazit

iTunes Connect ist die wichtigste Anlaufstelle für Entwickler nach Abschluss der Programmierung einer App oder eines App-Updates, und es ist schön zu sehen, dass Apple den Dienst pflegt und stetig weiterentwickelt. App Analytics wird zu einem immer mächtigeren und detaillierten Werkzeug, das Entwicklern dabei helfen kann, den Verkauf eigener Apps zu optimieren und potenzielle Lücken oder Schwachstellen im Vertrieb auszumachen. Da es darüber hinaus auch direkt mit dem eigenen Developer-Account gekoppelt ist, bedarf es keiner weiteren Konfiguration; die Daten zu den eigenen Apps können direkt abgerufen und eingesehen werden.

Zwar kommt die zugehörige iTunes-Connect-App für iOS (noch) nicht ansatzweise an den Funktionsumfang der Webplattform heran, fokussiert sich dafür aber auf die für Entwickler wohl interessantesten Informationen wie die Download-Zahlen und den Review-Prozess einer App. Beide Bereiche werden zukünftig weiter ausgebaut und optimiert.

Insgesamt sind die vorgestellten Neuerungen allesamt zu begrüßen und werden – mal mehr, mal weniger – das zukünftige Leben von Apple-Entwicklern erleichtern. ■

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

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, Philippe Bénard, Christian Bleske, Ekkehard Gentz, Thomas Hafen, Tam Hanna, Gerald Kammerer, Florian Laudahn, Bernhard Lauer, Florence Maurice, Ender Özgür, Michael Rohrlich, Jochen Schmidt, Markus Schraudolph, Katharina Sckommodau, Thomas Sillmann, Markus Stäuble

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

Mediaberatung und Content-Marketing-Lösungen
Thomas Wingenfeld
Telefon: (089) 741 17-124
sales@nmg.de

Klaus Ahlering
Telefon: (089) 741 17-125
sales@nmg.de

Stellenanzeigen / Anbieterverzeichnisse / Guides
Juliane Roschke
Telefon: (089) 741 17-283
sales@nmg.de

Disposition
Marita Brotz
Telefon: (089) 741 17-281
Fax: (089) 741 17-269
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

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

JURISTISCHE ASPEKTE DES PROGRAMMIERENS

Recht agil

Die Tätigkeit von Programmierern weist mehr juristische Aspekte auf, als man glauben mag.

So vielfältig der Arbeitsalltag von Programmierern ist, so verschieden sind auch die juristischen Probleme, die auftreten können. Deren Bewertung und auch die Lösungen hängen davon ab, ob es sich um einen festangestellten oder um einen freien Mitarbeiter handelt, wem welche Verwertungsrechte zustehen, wie die Arbeit organisiert ist und von vielen anderen Punkten mehr.

Agiles Arbeiten

Brainstorming, Konzeption, Beginn der Arbeit, Fertigstellung des Projekts, später gegebenenfalls noch Anpassungen – diese Arbeitsweise erscheint heutzutage vielfach antiquiert. Daher ist mittlerweile oftmals agiles Vorgehen das Mittel der Wahl. Was schon länger als Learning by Doing praktiziert wird und auch funktioniert, wird nun auch gerade in der Softwareproduktion angewandt.

Agile Software-Entwicklung kann sich zum einen auf einzelne Teilbereiche, andererseits aber auch auf den gesamten Prozess von Software-Entwicklung beziehen. Ziel ist dabei stets, bürokratischen Aufwand so weit wie möglich zu vermeiden. Es wechseln sich vielmehr kurze Planungs- beziehungsweise Entwicklungsphasen ab. Vorab werden lediglich die wesentlichen Ziele definiert und Prioritäten festgelegt. Nach der Erstellung eines Plans für die Version 1 beginnt auch schon die eigentliche Entwicklungsarbeit. Die agile Methode zeichnet sich durch einen indirekten, aber kurzen Weg aus. Nach dem Trial-and-Error-Prinzip wird neue Software in aller Regel möglichst früh eingesetzt und auf Verbesserungsmöglichkeiten abgeklopft.

Es existiert eine Vielzahl von unterschiedlichen agilen Methoden. Neben dem wohl bekanntesten agilen Rahmenwerk, Scrum, gibt es zum Beispiel auch noch folgende Ansätze:

- ActiF,
- Adaptive Software Development (ASD),
- Agile Model Driven Development (AMDD),
- Behavior Driven Development (BDD),
- Crystal,
- Design Driven Development (D3),
- Dynamic System Development Method (DSDM),
- Eclipse Way Process,
- Evolutionary Process For Integrating Cots-Based Systems (EPIC),
- Evolutionary Project Management and Product Development (EVO),
- Extreme Programming (XP),
- Feature Driven Development (FDD),
- Iconix,
- Internet-Speed Development,
- Lean Software Development (LSD),
- Microsoft Solutions Framework For Agile Software Development,
- Mobile-D,
- Rapid Application Development (RAD),
- Test Driven Development (TDD),
- Unified Process (UP),
- Agile Unified Process (AUP),
- Essential Unified Process (EssUP).

Damit agiles Arbeiten funktioniert, ist eine sehr enge und direkte Zusammenarbeit zwischen Programmierer und Auftraggeber notwendig. Agiles Vorgehen ist eher für umfangreiche beziehungsweise neue Projekte geeignet, kleine beziehungsweise bereits bestehende Projekte profitieren davon hingegen kaum.

Die vertragsrechtliche Einordnung agiler Software-Entwicklung ist leider nach wie vor stark umstritten. Die überwiegende Ansicht geht jedoch davon aus, hierbei als Grundlage ebenfalls das Werkvertragsrecht heranzuziehen. Die

Praxis-Tipp

Sowohl aus Sicht von Arbeitnehmern, Arbeitgebern und auch von Freelancern sind Lizenzverträge äußerst sinnvolle Instrumente. Darin sollten folgende Inhalte geregelt werden:

- Vertragsgegenstand / Leistungsumfang,
- Ziel / Einsatzgebiet,
- Umfang des Nutzungsrechts,
- Nutzungsdauer,
- Systemanforderungen,
- gegebenenfalls einzelne Projektabschnitte,
- Fragen der Haftung beziehungsweise Gewährleistung,
- Abnahmeregelungen,
- Entgelt / Zahlungsmodalitäten,
- gegebenenfalls Einsatz von Freelancern,
- Geheimhaltungspflicht,
- Dokumentation,
- Quellcode-Hinterlegung (Escrow),
- Installation / Schulung / Support / Wartung,
- Verwertungskette / Rechteübertragung auf Dritte.

Sofern keinerlei Regelungen getroffen werden, findet die sogenannte Zweckübertragungslehre Anwendung. Danach werden im Zweifel nur solche Rechtspositionen übertragen, die zur Erreichung des konkreten Vertragszwecks erforderlich sind.

agile Herangehensweise sieht jedoch keine klaren Grenzen vor, also etwa feste Start- und Endzeitpunkte. Aus diesem Grund sollte entsprechend den einzelnen Phasen (sogenannte Sprints) jeweils eine Zwischenabnahme erfolgen.

Nun gibt es bei agiler Software-Entwicklung kein vorab in allen Details festgelegtes Konzept, sodass hierbei auch die juristische Bewertung agil erfolgen muss. Dreh- und Angelpunkt sind in diesem Kontext ebenfalls die einzelnen Sprints. So betrachtet ist der entscheidende Maßstab die sach- und fachgerechte Arbeit beziehungsweise deren Ergebnisse. Neben diesen sollten unter anderem auch die Qualifikationen der beteiligten Programmierer sowie die grundlegenden Anforderungen und Ziele vertraglich fixiert werden.

Freelancer versus Festanstellung

Während das agile Arbeiten grundsätzlich im Rahmen von freier Mitarbeit, aber auch von Festanstellungsverhältnissen eingesetzt werden kann, gibt es in dieser Hinsicht diverse Punkte, die eine Differenzierung erforderlich machen. Denn es macht für alle Beteiligten einen Unterschied, ob es sich um den Abschluss eines Werkvertrags mit einem Freelancer oder um eine Arbeitsanweisung an einen Angestellten handelt.

In der Praxis ist nicht immer auf den ersten Blick erkennbar, wie der arbeitsrechtliche Status quo ist. Bei einem Angestellten müssen insbesondere folgende Kriterien erfüllt sein:

- gesetzliche Kündigungsfristen,
- Mindestanzahl von Urlaubstagen pro Jahr,
- enge Einbindung in die Betriebsorganisation,
- keine freie Zeiteinteilung bei der Arbeit,
- Vorgaben und Weisungen durch den Arbeitgeber,
- Verpflichtung zur persönlichen Leistungserbringung,
- Erbringung der ganzen/überwiegenden Arbeitskraft.

Natürlich kann im jeweiligen Arbeitsvertrag zu dem einen oder anderen Aspekt etwas Abweichendes vereinbart werden. Manche Dinge gibt das Gesetz jedoch zwingend vor, wie beispielsweise den Anspruch auf Urlaub oder Lohnfortzahlung im Krankheitsfall.

Im Unterschied dazu steht freien Mitarbeitern hauptsächlich der Anspruch auf Zahlung des vereinbarten Honorars zu. Ihre Auftraggeber müssen für sie insbesondere keine Sozialabgaben entrichten.

Für die Einordnung eines Programmierers als Arbeitnehmer oder freier Mitarbeiter kommt es nicht auf die Begrifflichkeit oder die Gestaltung des Vertrags an. Es ist vielmehr die objektive Situation maßgeblich. Wird jemand faktisch als Arbeitnehmer behandelt, muss er also etwa in den Räumlichkeiten des Auftraggebers arbeiten (anstatt im Homeoffice) und ist er weisungsgebunden, dann spricht schon einiges dafür, ihn eher als Angestellten anzusehen. Dies hat wiederum zur Folge, dass ihm Urlaub, Entgeltfortzahlung et cetera zu gewähren sind.

Auf dem Gebiet des Urheberrechts gibt es folgende Unterscheidung: Einem selbstständigen Programmierer steht das Urheberrecht an der von ihm geschaffenen Software zu, auch wenn er sie im Auftrag eines Dritten erstellt hat. Zwar gehen die Nutzungsrechte im vereinbarten Umfang auf den Auftraggeber über, der Programmierer ist und bleibt jedoch der originäre Urheber.

Im Rahmen eines Angestelltenverhältnisses sieht die Sache ein wenig anders aus. Wird die Software durch einen Arbeitnehmer erstellt, der in Wahrnehmung seiner arbeitsvertraglichen Aufgaben beziehungsweise nach den Anweisungen seines Arbeitgebers handelt, dann ist ausschließlich der Arbeitgeber zur Ausübung aller vermögensrechtlichen Befugnisse an der Software berechtigt – unabhängig davon, ob es sich um einen normalen Angestellten oder um einen Beamten beziehungsweise Angestellten im öffentlichen Dienst handelt.

Dieser Grundsatz gilt immer dann, wenn nichts anderes vereinbart wird. Die wesentlichen Nutzungs- beziehungsweise Verwertungsrechte gehen wegen des bestehenden Arbeitsvertrags auf den Arbeitgeber über. Allerdings lassen sich auch abweichende Regelungen vereinbaren. Dies muss dann jedoch ausdrücklich geregelt werden, im Idealfall natürlich schriftlich.

Das Ausgeführte gilt allerdings nicht für Software, die der Arbeitnehmer in Eigeninitiative erstellt hat. An diesen Werken behält er seine Rechte vollumfänglich. Die Abgrenzung zwischen freiem und Arbeitnehmer-Werk ist nicht immer leicht zu treffen. Es ist darauf abzustellen, ob und inwieweit ein enger innerer Zusammenhang zwischen den festgelegten Aufgaben des Programmierers und der betreffenden Software besteht. Dabei ist regelmäßig auf den individuellen Einzelfall abzustellen. Als Indizien kann etwa herangezogen werden, ob das Programm während der Arbeitszeit oder nach Feierabend erstellt wurde, ob es auf einem Dienst- oder auf einem privaten Computer geschrieben wurde und wie der Programmierer auf die Idee für die Software gekommen ist. ■

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.rechtssicher.info

ARBEITSMARKT

TRENDS UND JOBS FÜR ENTWICKLER

Weiterbildung

In die Köpfe investieren

Laut einer aktuellen Bitkom-Umfrage geben zwar 91 Prozent der Unternehmen an, dass Digitalkompetenz künftig mindestens so wichtig sein wird wie soziale und fachliche Kompetenz, zugleich bilden aber nur 36 Prozent der Unternehmen ihre Mitarbeiter weiter. Der Branchenverband hat auch nach den Gründen für die Zurückhaltung gefragt: Jedes dritte Unternehmen gibt an, dass ihm die Weiterbildungsangebote zu teuer sind. 31 Prozent beklagen, dass sie die Qualität der Angebote nicht beurteilen können. Jedem vierten Unternehmen fehlt ein Überblick, und 24 Prozent sagen, dass die vorhandenen Angebote nicht das abdecken, was in ihrem Unternehmen benötigt wird, oder dass sie auf die Arbeitskraft ihrer Mitarbeiter nicht verzichten können (23 Prozent).

»Ganz klar müssen wir mehr in die Köpfe investieren, wenn wir die digitale Transformation in Deutschland erfolgreich gestalten wollen«, sagt Bitkom-Präsident Thorsten Dirks. »Wir haben das Zeug dazu, in der digitalen Wirtschaft eine ähnliche Führungsrolle zu erreichen, wie wir sie bislang in der Automobilindustrie und dem Maschinenbau haben. Aber wir müssen das Potenzial nutzen. Innovationen entstehen durch die Mitarbeiter in den Unternehmen – und Innovationen sind es, die den wirtschaftlichen Erfolg Deutschlands ausmachen.«

Jobs für Web- und Mobile-Entwickler

Bayern, NRW und Baden-Württemberg sind die Bundesländer, für die derzeit die meisten Angebote für Webentwickler ausgeschrieben sind (Bild 1). Ein Blick auf die Auswertung der Großstädte anhand der Datenbank von Jobkralle.de zeigt, dass besonders viele Angebote für München (770 Treffer) und Berlin (711 Treffer) ausgeschrieben sind. Auch in Hamburg (531 Treffer) und Frankfurt am Main (531 Treffer) werden aktuell viele Webentwickler gesucht.

Bei der Auswertung für Mobile-Entwickler steht Berlin ganz oben. 460 Treffer lieferte die Datenbank hier auf der Suche nach Jobangeboten. Auf dem zweiten Platz liegt Hamburg (323 Treffer), gefolgt von München (317 Treffer). Der Vergleich der einzelnen Bundesländer zeigt dann wieder Bayern ganz vorne, vor Berlin, das sich hier aber noch vor Nordrhein-Westfalen platzieren konnte (Bild 2).

In aktuellen Jobangeboten am häufigsten gesucht sind Kenntnisse in den Programmiersprachen Java, C++, JavaScript und HTML. Die Sprache C# steht mit 10,7 Prozent auf dem fünften Rang.

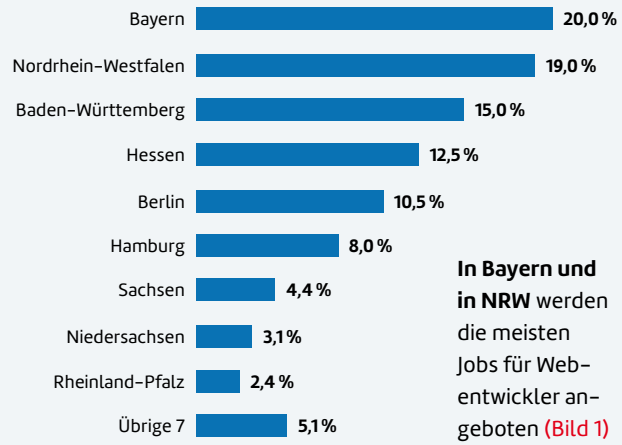
Ein Blick auf die in Tabelle 1 gelistete Nachfrage nach Technologiekenntnissen zeigt, dass iOS in diesem Monat stärker nachgefragt ist. Es kletterte von Platz 7 im Vormonat auf Platz 5. Zurückgefallen sind dafür Big Data und Android.

Tabelle 1: Technologien

Rang	Technologie	Anteil *
1	Cloud	19,4 %
2	MySQL	11,0 %
3	HTML5	9,7 %
4	SharePoint	8,4 %
5	iOS	7,7 %
6	Big Data	7,2 %
7	Android	6,5 %
8	Microsoft SQL Server	6,1 %
9	CSS3	4,6 %
10	Windows 10	4,0 %
11	AngularJS	4,0 %
12	NoSQL	2,7 %
13	WPF	2,6 %
14	Responsive Web	2,6 %
15	Azure	2,1 %
16	WCF	1,4 %

* Prozentualer Anteil der Treffer

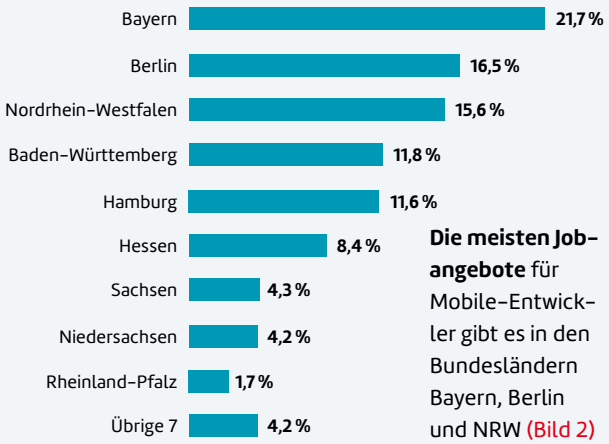
Webentwickler: Ländervergleich



In Bayern und in NRW werden die meisten Jobs für Webentwickler angeboten (Bild 1)

web & mobile developer 11/2016 Quelle: Eigene Erhebungen, Jobkralle.de

Mobile-Entwickler: Ländervergleich



Die meisten Jobangebote für Mobile-Entwickler gibt es in den Bundesländern Bayern, Berlin und NRW (Bild 2)

web & mobile developer 11/2016 Quelle: Eigene Erhebungen, Jobkralle.de

Zahl des Monats

Die am schnellsten wachsende ITK-Nation ist Indien mit einem Plus von **6,7 Prozent** auf prognostizierte 64 Milliarden Euro Jahresumsatz 2016, knapp vor der Türkei (plus **6,1 Prozent** auf 20 Milliarden Euro). Weltweit größter ITK-Markt bleiben die USA: plus **3,5 Prozent** auf 956 Milliarden Euro.

Quelle: EITO

Solcom

Abgeschwächte Dynamik

Die Entwicklung im Projektmarkt ist im Vergleich zu den vergangenen Jahren leicht rückläufig, bewegt sich aber noch immer auf sehr hohem Niveau (Bild 3). Das sind die Ergebnisse der aktuellen Solcom-Marktstudie. Zudem haben bei Freiberuflern die Stundensätze wieder die höchste Priorität (Bild 4). Ein Großteil der befragten Freiberufler arbeitet mehr als 40 Stunden pro Woche, jeder Vierte sogar mehr als 50. Dennoch empfindet über die Hälfte ihre Tätigkeit als gut mit dem Familien- beziehungsweise Privatleben vereinbar, nur jeder Fünfte macht dabei andere Erfahrungen. So sieht auch die Mehrheit der Umfrageteilnehmer ihre Selbstständigkeit als besser vereinbar mit dem Privatleben als eine Festanstellung, Ein Drittel sieht dies jedoch genau umgekehrt. Anders als im Vorjahr ist für die befragten Freiberufler nicht mehr die Aufgabe das wichtigste Projektkriterium, sondern der Stundensatz. Gleichzeitig sieht eine absolute Mehrheit im laufenden Jahr Stundensätze stagnieren, steigende Stundensätze erwarten weniger Umfrageteilnehmer als im vergangenen Jahr. Für das Gesamtjahr geht zwar noch immer eine Mehrheit von einer Verbesserung der Situation oder zumindest einem konstanten Niveau aus, im Vergleich zur Vorjahresbefragung ist der Trend jedoch rückläufig.

Fraunhofer

Digitales Lernen mit Roberta

Auf der Young IFA 2016 in Berlin konnten die Besucher des Fraunhofer-Programmierlabors Roboter zum Leben erwecken. Seit 14 Jahren nutzt die Roberta-Initiative die Faszination von Robotern, um Jungen und Mädchen fit für Technik zu machen. In den Kursen programmieren sie selbst gebaute Maschinen. Mit bisher mehr als 350.000 Schülerinnen und Schülern und über 1000 zertifizierten Roberta-Teachern fördert das MINT-Bildungsprogramm des Fraunhofer IAIS digitale Bildung.

Die frei verfügbare Open-Source-Online-Programmierumgebung Open Roberta Lab ermöglicht es jedem, der einen Computer, Tablet oder Smartphone mit Internetzugang hat, das Programmieren zu erlernen. Die grafische Fraunhofer-Programmiersprache NEPO macht selbst aus Anfängern im Handumdrehen Entwickler.

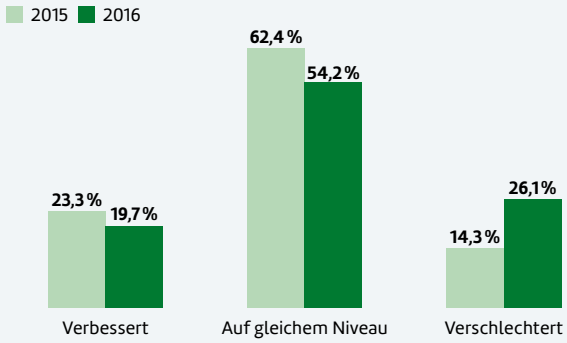
Bitkom

Start-ups als Jobmotor

Gründer aus der Digitalbranche schaffen in Deutschland Tausende Arbeitsplätze. Im Durchschnitt beschäftigt jedes Start-up 15 Mitarbeiter – 2015 wurden durchschnittlich nur 13 Jobs gezählt. Das ist das Ergebnis einer Bitkom-Umfrage. Sechs von zehn Start-ups (58 Prozent) geben an, dass sie im vergangenen Jahr neue Mitarbeiter eingestellt haben, nur vier Prozent mussten Personal abbauen. »Start-ups sind nicht nur von entscheidender Bedeutung für das Gelingen der digitalen Transformation in Deutschland, sie sind auch ein bedeutender Arbeitgeber und Wirtschaftsfaktor«, sagt Bitkom-Geschäftsleiter Niklas Veltkamp.

Auch in diesem Jahr wollen knapp drei Viertel der Start-ups Personal einstellen. Nur ein Prozent der Gründer plant einen Stellenabbau. Nicht immer finden Start-ups allerdings geeignete Kandidaten. Jeder vierte Gründer nennt einen Mangel an Fachkräften als gravierendes Hemmnis. Veltkamp: »Viele Start-ups würden sogar gerne mehr Leute einstellen. Doch gerade bei Top-Programmierern und anderen Digital-Experten übersteigt die Nachfrage das Angebot – und Start-ups müssen Kandidaten gewinnen, die auch von großen Konzernen mit ganz anderen Möglichkeiten umworben werden.«

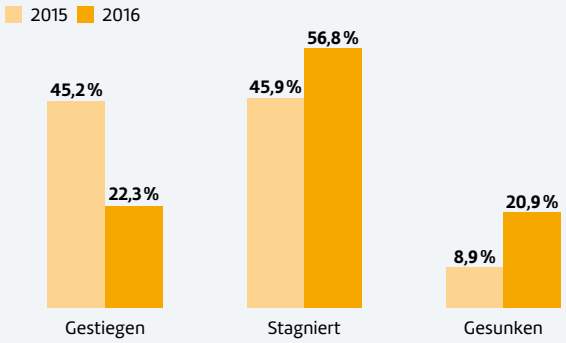
Projektauslastung 2016



Der Anteil der Freiberufler, deren Projektauslastung sich verbessert hat, ist leicht zurückgegangen. Jeder Vierte sieht im laufenden Jahr eine schlechtere Projektauslastung (Bild 3)

web & mobile developer 11/2016 Quelle: Solcom Marktstudie Zwischenbilanz 2016

Stundensätze 2016



Jeder fünfte Freiberufler meldet, dass die Stundensätze gesunken sind. Die deutliche Mehrheit sieht stagnierende Sätze (Bild 4)

web & mobile developer 11/2016 Quelle: Solcom Marktstudie Zwischenbilanz 2016

dotnetpro Newsletter

Top-Informationen für den .NET-Entwickler.
Klicken. Lesen. Mitreden.

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 - Software für Marktführer
Intelligente Software für internationale Konzerne und mittelständische Unternehmen: prodot stärkt seit über 15 Jahren namhafte Kunden im weltweiten Wettbewerb – mit effizienten, stabilen und kostensenkenden Lösungen.
Kunden schätzen unsere Kreativität. Mit Sorgfalt und Enthusiasmus entwickeln wir hochwertige Software. Digitale Prozesse und innovative Technologien sind unser Antrieb, Fortschritt und Kontinuität unser Anspruch.
ALDI SÜD, Microsoft und Siemens vertrauen uns bereits viele Jahre.
Gerne zeigen wir Ihnen warum! Sprechen Sie mich an. 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

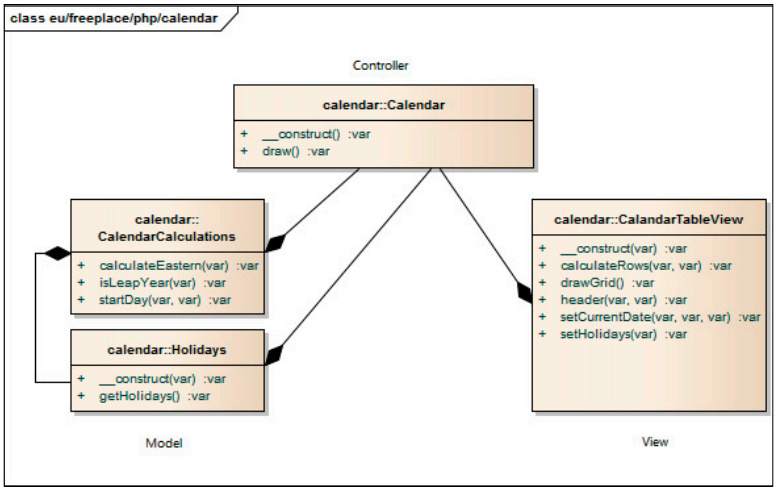
PAYONE ist einer der führenden Payment Service Provider und bietet modulare Lösungen zur ganzheitlichen Abwicklung aller Zahlungsprozesse im E-Commerce. Das Leistungsspektrum umfasst die Zahlungsabwicklung von allen relevanten Zahlarten mit integriertem Risikomanagement zur Minimierung von Zahlungsausfällen 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

Die Ausgabe 12/2016 erscheint am 10. November 2016

Objektorientierte Programmierung mit PHP



Mit fertigen Bibliotheken zu arbeiten, um bei der Auftragsbewältigung konkurrenzfähig zu bleiben, ist legitim und durchaus üblich. Wenn allerdings für das vorhandene Problem noch keine Lösung existiert, besteht die Notwendigkeit, die Aufgabe in Eigenregie zu bewältigen. Wie bei einer solchen Herausforderung eine mögliche Lösung aussehen kann, zeigt die Konzipierung und Realisierung eines Kalenders in PHP. Da die Funktionsweise von Kalendern allgemein bekannt ist und die Fachlichkeit hinreichend komplex ist, ist dies eine naheliegende Wahl. Dieser Workshop zeigt, wie man im objektorientierten Umfeld Probleme modelliert und in PHP implementieren kann. Und er zeigt, wie man mit solidem Basiswissen und gut überlegtem Vorgehen fast jedes Problem bewältigen kann.

Gamification als Motivator

Gute Spiele motivieren von selbst und machen Spaß. Das gilt gleichermaßen für Computerspiele, Brettspiele oder jede andere Form von Zeitvertreib. Gamification ist der Versuch, die Motivatoren eines Spiels auf spielfremde Bereiche zu übertragen. Das Ziel: Nutzer sollen zu unliebsamen oder nervigen Aufgaben motiviert werden. Auch im Bereich der Programmierung von mobilen Anwendungen hat dieser Ansatz Einzug gehalten.

Apps für Nachrichten

Es ist wohl mit das größte Highlight in der neuen Version 10 von iOS: die stark überarbeitete Nachrichten-App. Sie ist bunter, vielseitiger, erlaubt mit sogenannten Stickern neue Interaktionsmöglichkeiten und bietet interessante Features wie versteckte Nachrichten, die erst lesbar werden, wenn der Empfänger über die Nachricht wischt, oder das Übertragen von Stimmungen durch Aufploppen oder vorsichtiges Einfliegen von Nachrichten beim Empfänger.

Das Selection API

Über das Selection API kann man Textinhalte auf einer Webseite per JavaScript auswählen und bearbeiten. Das API befindet sich momentan beim W3C im Status Working Draft, wird mittlerweile aber von den meisten modernen Browsern unterstützt. Das API definiert das Interface Selection, das die aktuelle Auswahl auf einer Webseite repräsentiert, und erweitert die Interfaces Document und Window um die Methode *getSelection()*.

dotnetpro

Ausgabe 11/2016 ab 20. Oktober 2016 am Kiosk
Mit den Tools und Frameworks von Xamarin schreibt man Apps für iOS oder Android in C#. Grundlage bildet das Mono-Projekt, das .NET auf andere Plattformen portiert und von Xamarin weiterentwickelt wird. Der dotnetpro-Schwerpunkt dreht sich um dieses Thema.
www.dotnetpro.de

Unsere digitalen Angebote

Wöchentlicher Newsletter
webundmobile.de/newsletter

Stellenmarkt
stellenmarkt.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:
Klaus Ahlering, Thomas Wingenfeld • Tel. 089/74117-125 • sales@nmg.de

DWX Developer Week 2017

26.-29. Juni 2017, Messe Nürnberg

- **Treffen Sie** Ihre Zielgruppe auf einer der größten Entwickler-Konferenzen Europas
- **Profitieren Sie** von umfangreichen Marketingaktivitäten
- **Gestalten Sie** das Progamm selber mit und nehmen Sie am Call for Papers teil

developer-week.de/Ausstellung

Diese Kunden vertrauen uns:

adorsys · AIT · brainLight. LIFE IN BALANCE · best-blu consulting with energy · cellent. ...more than just IT · CENTIGRADE USER INTERFACE ARCHITECTURES · centron · codecentric · DataAccess EUROPE · DATEV · DevExpress · ECS WEBHOSTING · embarc Software Consulting GmbH · embarcadero · ERGOSIGN · evosoft · HANSER · HeiReS · ILNUMERICS Computing for Applications · InnoGames · intel Software · INTERNETONE AKTIENGESELLSCHAFT · it-agile · MATHEMA · MINERVA · Microsoft · mitp · redgate ingeniously simple · reddo SOLUTIONS · SafeNet · SDX · sevenval · SIGS DATACOM · SOS SOFTWARE SERVICE VALUE ADDED DISTRIBUTION · THEOBALD SOFTWARE · UNIRENT EDV-Systemtechnik GmbH

developer-week.de **#dwx17** **DeveloperWeek**