

# Fachartikel

## Docker Container und deren Verwendung im Webbereich

**Modulnummer:** *DPL4000*  
**Modulname:** *Fachartikel*  
**Abgabedatum:** *10. April 2020*  
**Semester:** *März 2020*  
**Name:** *Noelia Heinrich Chicharro*  
**Campus:** *Zürich*  
**Land:** *Schweiz*  
**Wortanzahl:** *+17'000*

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

10. April 2020  
Ort, Datum



Unterschrift Student

## Wozu braucht es Container?

Wenn mehrere Entwickler an derselben Anwendung arbeiten, war es in der Vergangenheit üblich, dass man die notwendige Entwicklungsumgebung auf einem physischen Computer oder einer Virtuellen Maschine bereitstellt.

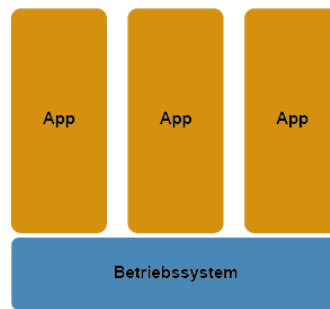
Im ersten Fall wird ein Computer mit einem Betriebssystem und den notwendigen Applikationen versehen.

Im zweiten Fall wird auf einem Gerät, welches ein Betriebssystem hat, ein weiteres Betriebssystem installiert. Auf dessen werden die benötigten Anwendungen installiert.

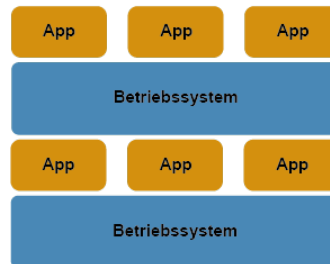
Virtualisierte Plattformen eröffnen der IT neue Möglichkeiten im Bereich der Kostensenkung. Da mehrere Systeme auf einem physischen Gerät laufen können.

Jede VM simuliert ein komplettes Gerät mit Betriebssystem, BIOS, eigenem Arbeitsspeicher und so weiter. Dies wiederum benötigt viel Systemspeicherplatz des tatsächlichen Computers. Dies hat zur Folge, dass es relativ lange dauert bis solche Systeme bereit sind. Um dies etwas im Zaum zu halten, ist es möglich die verschiedenen VM's anzupassen und deren

Bare-Metal-System



Virtuelle Maschine



Softwareressourcenverbrauch etwas zu regulieren. Ab einem gewissen Punkt bringt dies leider auch nichts mehr.

Der Nachteil dieser beiden Varianten ist aber vor allem, dass die Flexibilität mangelhaft ist. Die Aktualisierung der Anwendungen ist aufwändig und zeitintensiv. Zudem lassen sich solche Umgebungen schlecht auf andere Geräte portieren und müssen umständlich synchronisiert werden.<sup>1</sup>

Aus diesem Grund wurden Container entwickelt.

## Was ist ein Container?

Streng genommen ist ein Container auch nur eine virtuelle Maschine. Das Hauptmerkmal, in welchem sie sich unterscheiden ist jedoch, dass ein Container kein eigenes Betriebssystem hat.

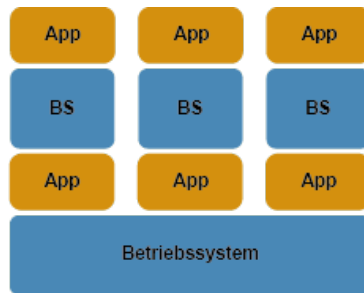
### Was heisst das nun genau?

Während die VM ein komplettes physisches Gerät simuliert, fängt die Abstraktion des Containers erst auf der Applikationsebene an. Somit benutzt der Container das schon vorhandene Betriebssystem. Dadurch dass diese Vorgehensweise weniger Ressourcen verbraucht ist es möglich mehr

---

<sup>1</sup> (Haluschak, 2019)

Anwendungen zu speichern und schneller zu booten.<sup>2</sup>

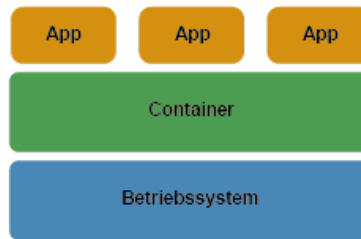


### Wie entsteht ein solcher Container ohne, dass er ein Betriebssystem emuliert?

Im Grunde ist dies dank Container-Images möglich. Ein solches Image besteht aus einer Software und allen Abhängigkeiten, welche es zum Laufen benötigt. Zudem können, auf selbigem zusätzliche Applikationen eingebaut werden.

Ich persönlich stelle mir ein solches Image als Collage vor. Viele kleine Teile die am Ende ein Gesamtes ergeben.

Diese Container-Image können auch von anderen abgeleitet und weitergeführt werden. In diesem Sinne spricht man von einem Abbild. Zusätzlich zu den genannten Inhalten, befinden sich Programmcode, Datenbanken, Bibliotheken und all deren Abhängigkeiten in einem solchen Image. Über die Konsole wird das gesamte Image, zum benötigten Zeitpunkt, gestartet.<sup>3</sup>



### Was sind nun die konkreten Vorteile von Containern?

Zum einen liegen die Vorteile eines Containers bei der Ressourceneinsparung. Wie schon erwähnt müssen keine zusätzlichen Maschinen zur Verfügung gestellt werden, auch weitere Betriebssystemlizenzen fallen weg. Dies ist sowohl im privaten als auch im kommerziellen Gebrauch von grossem Vorteil. Ein Punkt, welcher vor allem im geschäftlichen Sinne grosse Relevanz hat, ist derjenige, dass Container extrem flexibel sind. Da die komplette Arbeitsumgebung auf einem Container erstellt werden kann, ist ein Mitarbeiter nicht an eine physische Maschine gebunden. Der Angestellte kann nun, beispielsweise, auch von zu Hause auf die Datenbank zurückgreifen, sobald er den Container laufen lässt ist diese von überall her nutzbar.

## Docker

Docker ist eine der grössten Entwickler im Bereich von Containern. Sie haben es sich zum Ziel gemacht, das Leben eines jeden Entwicklers zu vereinfachen und dessen Flexibilität zu fördern.

### Kurze Geschichte von Docker

Im Jahre 2013 gründete die Firma dotCloud den Ableger Docker.

Noch im selben Jahr gab dotCloud bekannt, dass sie die bestehende Firma in Docker Inc. umbenennen würden. Von da an gab es kein Halten mehr.

Nur wenige Monate später wurde die 'Platform-as-a-Service' Funktion an eine Berliner Firma namens cloudControl verkauft. Von diesem Zeitpunkt an gewann Docker immer mehr Aufsehen. Soviel dass sie noch im selben Jahr Teil des Red Hat Enterprise Linux 7.0 wurden. Es folgten viele weitere Kooperationen.

Unter anderem schlossen sie sich, mit anderen Firmenzusammen, dem Kubernetes-Projekt von Google an.

Anfänglich investierten verschiedene Geldgeber rund 15 Millionen in Docker, damals noch dotCloud. Im darauffolgenden Jahr fanden sich noch mehr Investoren, diese brachten

<sup>2</sup> (Docker, kein Datum)

<sup>3</sup> (Augsten, 2018)

weitere 95 Millionen. Mit den insgesamt 120 Millionen erreichte Docker das, was sie heute sind.

“Our success relies upon our people and our culture... these virtues provide our foundation of who we are how we make decisions.” - Docker

2019 verkaufte man Docker an Mirantis, diese gaben bekannt, dass der Support des Docker Swarms innert der folgenden zwei Jahre eingestellt wird.<sup>4</sup>

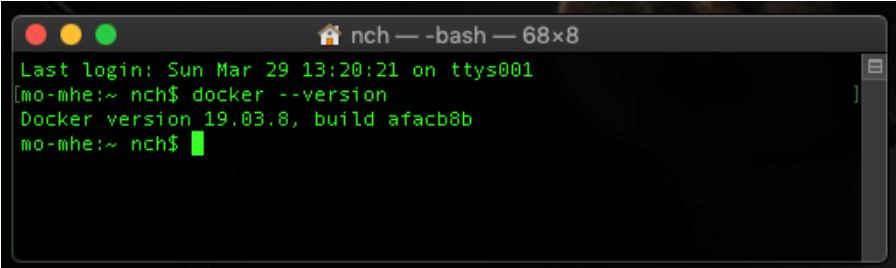
## Die Problematik

Während immer mehr Applikationsentwickler auf Container zurückgreifen, um ihre Arbeit produktiver zu gestalten. Gibt es viele Webentwickler, die von Docker oder deren Containern noch nie etwas gehört haben. Viele Webentwickler haben ihre Entwicklungsumgebung fest auf ihrem Arbeitscomputer installiert. Doch was wenn dieser nicht mehr funktionsfähig ist, oder der Entwickler seine Arbeit jemanden weiter geben muss? Hier besteht, meiner persönlichen Meinung nach, eine grosse Lücke, die dringend gefüllt werden muss. In folgendem werde ich, in einem Beispiel, aufzeigen wie man als Webentwickler

in den Genuss der Docker Container kommen kann. Zudem werde ich zusätzlich nochmals alle Vor- und Nachteile dessen genauer darstellen. Das Beispiel wird aufzeigen, wie man eine gesamte Webentwicklungsumgebung auf einem Container aufsetzen kann.

Nun hat man die Gewissheit, dass Docker richtig installiert wurde. Doch läuft es auch so wie es sollte? Auch für diesen Fall gibt es einen einfachen Test, welcher zu Beginn empfohlen wird. Docker bietet ein einfaches ‘hello-world’ Image an, welches laufen gelassen werden kann.

## Docker installieren auf eigenem Gerät

A screenshot of a terminal window on a macOS system. The window title is 'nch — -bash — 68x8'. The terminal shows the command 'docker --version' being executed, which returns 'Docker version 19.03.8, build afacb0b'. The prompt 'mo-mhe:~ nch\$' is visible at the bottom.

```
nch — -bash — 68x8
Last login: Sun Mar 29 13:20:21 on ttys001
mo-mhe:~ nch$ docker --version
Docker version 19.03.8, build afacb0b
mo-mhe:~ nch$
```

Docker Container laufen von Haus aus auf dem Linux Betriebssystem. Da die Mac-Betriebssysteme auf dessen basieren ist es ein leichtes die Installation zu adaptieren. Für Windowsbenutzer gibt es mittlerweile auch eine einfache Lösung. Da ich selbst auf einem Applegerät arbeite, wird mein kommendes Beispiel die MacOS Variante abdecken. Docker wird wie die meisten Applikationen heruntergeladen und installiert. Um festzustellen, ob die Installation erfolgreich war wurde über die Konsole nach der Version gesucht. Erscheint diese, ging alles gut.

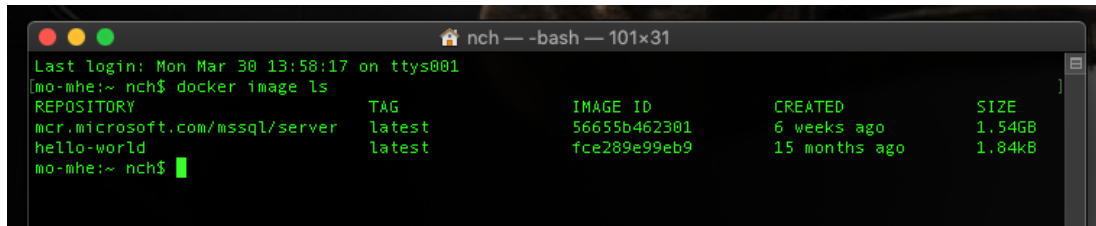
Dieser Befehl hat nun verschiedene Dokumente heruntergeladen, welche das Image vervollständigen. Um die Struktur von Beginn an zu verstehen suchen wir mal danach.

Jetzt können wir uns sicher sein, dass Docker richtig und vollständig installiert wurde.

An diesem Punkt angelangt, stellt man sich die Frage was man den genau braucht, um eine ideale Webentwicklungsumgebung aufzubauen. Jedem Webentwickler werden Begriffe wie Apache, MySQL und PHP geläufig sein. Daher werden in den folgenden Kapiteln genau selbige auf einen

---

<sup>4</sup> (plenty, 2020)



```
nch — -bash — 101x31
Last login: Mon Mar 30 13:58:17 on ttys001
mo-mhe:~ nch$ docker image ls
REPOSITORY              TAG                IMAGE ID           CREATED            SIZE
mcr.microsoft.com/mssql/server  latest            56655b462301      6 weeks ago       1.54GB
hello-world              latest            fce289e99eb9      15 months ago     1.84kB
mo-mhe:~ nch$
```

Container installiert und demonstriert wie diese verwendet werden können.

## Installation Apache Image in einem Container

Um eine Webseite testen zu können, benötigt es einen lokalen Server, welche über den Lokahost darstellt, was man entwickelt hat. In den meisten Fällen wird dies über Apache gemacht. Um das Leben vieler einfacher zu gestalten, gibt es schon vorgefertigte Apache Images im sogenannten Docker Hub.

Um das Image zu installieren, gibt es zwei Wege. Im ersten Fall wird ein Docker File im Projekt erstellt. Um den Rahmen dieses Artikels nicht zu sprengen, werden wird dies aber nicht machen.

‘Docker Hub is the world’s easiest way to create, manage, and deliver your teams container applications.’

Denn es gibt auch einen Weg dies, ohne ein eigenes Docker File zu kreieren. Manchmal will man schlichtweg auch kein zusätzliches Dokument in seinem Projekt haben.

Erst muss natürlich festgestellt werden, ob Docker überhaupt läuft. Dies kann über die Konsole oder das Docker Icon ein Im Terminal begibt man sich erst einmal auf das Projekt. Da werden folgende Befehle ausgeführt:

```
docker run -dit --name my-apache-app -p 8080:80 -v "$PWD":/usr/local/apache2 /htdocs/ httpd:2.4
```

Dieser Befehl zieht das gesamte Image aus der Docker Hub auf das Gerät.

Zur Kontrolle kann man im Docker Dashboard nachschauen ob der Container auch wirklich installiert wurde.<sup>5</sup>

Welchen Zweck hat dies nun genau? Mit dem, soeben installierten Apache-Server-Image ist es uns möglich

unser Projekt lokal auf dem Browser anzeigen zu lassen. Der Server wird zwar wie gewohnt separat gestartet.

Jedoch ist er an unser Projekt gebunden und frisst somit weniger Ressourcen. Möchte man in einem weiteren Projekt einen Apache-Server laufen haben, muss man diesen wieder individuell installieren. Es muss jedoch beachtet werden, dass dieses Image keinerlei Sicherheit mit sich bringt. Um mehr Sicherheit garantieren zu können, müssten weitere Installationsschritte begangen werden. Zudem, wie den meisten Webentwickler bekannt sein dürfte, müssen hierfür zusätzliche Lizenzen erworben werden, um Sicherheit gewährleisten zu können.

## Installation MySQL Image in einem Container

Neben all den statischen Webseiten tummeln sich auch viele dynamische Seiten in den Tiefen des Internets. Diese dynamischen Webseiten sind oft auch an eine Datenbank gebunden. Sei

<sup>5</sup> (Company, kein Datum)

```

[mo-mhe:projekt nch]$ docker run -dit --name my-apache-app -p 8080:80 -v "$PWD":/usr/local/apache2/htdocs/ httpd:2.4
Unable to find image 'httpd:2.4' locally
2.4: Pulling from library/httpd
c499e6d256d6: Pull complete
76155f771be0: Pull complete
48b718b71719: Pull complete
d65ae7a4c211: Pull complete
8d17dee838ad: Pull complete
Digest: sha256:13aa010584cb3d79d66adf52444494ae5db67faa28d65a1a25e6ddc57f7c0e2a
Status: Downloaded newer image for httpd:2.4
9cb88bc473cdd9f57808ccdda2d8feb48f5cbde614186c5d5fbd2b599936453e

```

es nun eine welche mehrere Benutzer sammelt oder Produkte eines Online-Shops beinhaltet. Am geläufigsten ist hier MySQL.

Die Installation des MySQL Images gestaltet sich ähnlich, wie die des Apache-Server-Images. Wobei es zu empfehlen ist, selbiges schon vor installiert zu haben. Um einen sauberen Arbeitsverlauf zu garantieren.

Wie schon im vorherigen Beispiel sollte zu Beginn festgestellt werden, ob Docker überhaupt läuft. Danach begibt man sich in der Konsole auf das gewünschte Projekt. Um sich keine Sorgen über die Version zu machen, gibt es einen Befehl, der automatisch die neueste Version für uns raussucht und genau diese dann auch auf unser Gerät zieht. Dies geschieht mit folgendem Befehl:

```

docker run -p 3306:3306 -d --name mysql -e MYSQL_ROOT_PASSWORD=password mysql/mysql-server

```

Nun hat unser Projekt Zugriff auf MySQL. Leider

bringt uns dies alleine noch nicht allzu viel. Um die komplette MySQL

Funktionalität zu erlangen, müssen weitere Schritte begangen werden.

Um wirklich auf die Datenbanken zugreifen zu können, ist eine MySQL-Workbench unverzichtbar. Jeder gute Webentwickler hat diese jedoch schon installiert.

In selbiger wird nun eine neue Verbindung geöffnet und unsere Daten eingetragen. Wichtig zu beachten ist nur, dass nicht der Root-Account vermerkt wird, sondern derjenige welcher im Image erstellt wurde.

Auch hier sollte bemerkt werden, dass keinerlei Sicherheit gewährleistet wird. Um diese sicherzustellen sind weitere Schritte von Nöten. Für eine lokale Benutzung ist dies aber vollkommen ausreichen.<sup>6</sup>

## Installation von PHP in einem Container

Generell nimmt man an, dass ein guter Webentwickler PHP global auf seinem Gerät installiert hat. Was ist aber nun wenn dieser die Maschine wechseln muss oder das Projekt an einen Kollegen,

ohne PHP, weitergeben muss?

Um diese Eventualität abzudecken empfiehlt es sich PHP direkt auf dem Projekt als Container zu installieren, um sicher zu gehen, dass es immer und überall lauffähig sein wird.

Hier bietet Docker Hub wieder eine tolle Image Vorlage, welche ohne Lizenz und weitere Kosten installiert werden kann. Der Ablauf ist praktisch identisch wie bei den Vorherigen: Sicherstellen, ob Docker läuft, in der Konsole zu dem gewünschten Projekt navigieren und folgenden Befehl eingeben.

```

docker run -it --rm --name my-running-script -v "$PWD":/usr/src/myapp -w /usr/src/myapp php:7.4-cli php your-script.php

```

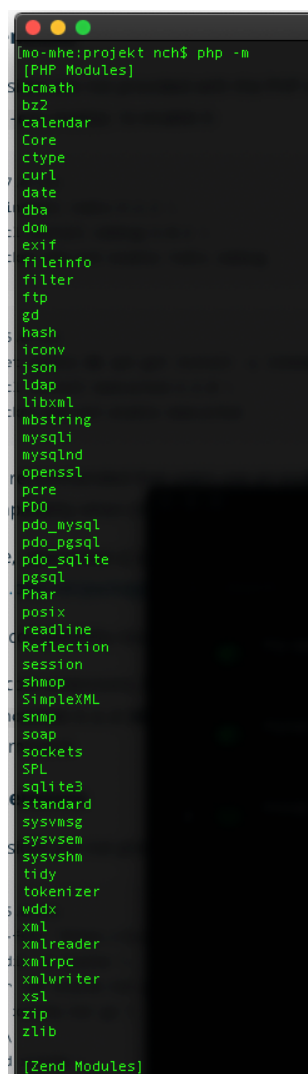
Nun hat man die einfachste PHP Variante im Projekt. Wer sich gut auskennt und weitere Extensions importieren will ist hier aber frei.

Diese kann man direkt anhand der PHP-Version beziehen. Jedoch darf nicht vergessen werden, dass jede zusätzliche Funktionalität eine Abhängigkeit hat, welche von Hand verzeichnet werden muss. Das reine Herunterladen wird hier nicht ausreichen. Ich werde

<sup>6</sup> (Chandrasekaran, 2017)

diesbezüglich, aber nicht genauer in das Detail gehen und dies den mutigen Webentwicklern selbst überlassen.

Zu unserem Glück bringt jede PHP-Version jedoch immer schon einige Funktionalitäten von Haus aus mit. Um diese einzusehen benutzt man wieder die Konsole. Sich im Projekt befindend, gibt man 'php -m' als Befehl ein und man erhält eine Liste mit allen Funktionalitäten, die ohne eigenhändige Abhängigkeit eintippen verwendbar sind.<sup>7</sup>



```
[mo-mhe:projekt nch$ php -m
(PHP Modules)
bcmath
bz2
calendar
Core
ctype
curl
date
dba
dom
ext
fileinfo
filter
ftp
gd
hash
iconv
json
ldap
libxml
mbstring
mysqli
mysqlnd
openssl
pcre
PDO
pdo_mysql
pdo_pgsql
pdo_sqlite
pgsql
Phar
posix
readline
Reflection
session
shmop
SimpleXML
snmp
soap
sockets
SPL
sqlite3
standard
sysvmsg
sysvsem
sysvshm
tidy
tokenizer
wddx
xml
xmlreader
xmlrpc
xmlwriter
xsl
zip
zlib
[Zend Modules]
```

Nun sollte alles was für eine perfekte Webentwicklungsumgebung von Nöten ist, installiert und bereit sein.

## Sicherheit eines Containers

Leider hat ein Container nicht nur Vorteile. Den auch obwohl Docker und andere Anbieter immer mehr an Beliebtheit gewinnen, gibt es im Bereich der Sicherheit eine grosse Lücke.

Da sich Docker Container denselben Kernel teilen, wie die Maschine auf welcher er installiert wurde, verbraucht er nur wenig Platz und verschwendet keine Ressourcen. Im Normalfall kann ein Angreifer nicht auf den Kernel des Hosts zugreifen und somit nicht allzu grossen Schaden anrichten. Ist aber ein Container an diesen gekoppelt sieht es leider etwas anders aus. Bekommt der Angreifer Zugriff auf den Container kann er somit auch auf den Kernel des Hosts zugreifen und so viele Subsysteme ansprechen und einsehen. Er umgeht also die Sicherheitseinstellungen des Gerätebesitzers. Ein weiteres Problem ist, dass der Container oft mit dem Standard-User-Namespace versehen wird. Somit erlangt der Angreifer Admin-Rechte auf dem gesamten Gerät.

Um dies zu verhindern gibt es aber eine relativ einfache Lösung. Der Container kann so konfiguriert werden, dass er nicht auf alle Rootrechte zugreifen kann. Somit wird wenigstens ein bisschen mehr Sicherheit gewährleistet.

Nun stellt sich natürlich die Frage wie man sich dagegen wehren kann. Gibt es schon eine Lösung?

Die neueren Versionen von Docker beschäftigen sich etwas damit. Sie besitzen Funktionalitäten, welche die Angriffe etwas abschwächen. Zum Beispiel konvertieren sie einige Rootrecht-Dokumente in Read-Only, damit der Angreifer diese nicht manipulieren kann. Dazu gehören Dinge wie Netzwerkconfiguration oder Kernelmodul Ladeabläufe. Einsehbar sind sie aber trotzdem noch. Docker bestätige, dass sie in Zukunft an der Sicherheit ihrer Container arbeiten werden.

Ihr momentan grösstes Ziel ist es, die Rootinformationen von Angreifern zu schützen. Die Schwierigkeit besteht jedoch darin die Dokumente so zu sichern, dass sie nicht mehr angreifbar sind, aber so frei zulassen, dass sie noch in der Lage sind gemeinsam zu kommunizieren. Viele Entwickler befürchten daher, dass sie nach den Sicherheitsanpassungen,

<sup>7</sup> (Company, 2020)

weniger Zugriff auch die genannten Dokumente haben werden.

Am Ende muss man aber anmerken, dass sich das Team von Docker sichtlich Mühe gibt bestehende Fehler oder Lücken zu beheben. Sie reagieren schnell auf Stimmen aus der Community und entwickeln sich stetig weiter.<sup>8</sup>

### **Allgemeines Fazit**

Generell bin ich persönlich der Meinung, dass Docker mit ihren Containern auf einem guten Weg sind die Entwicklungsbranche zu revolutionieren.

Die Möglichkeiten für Webentwickler steigern sich immens, in Bereichen der Flexibilität und Produktivität.

Jedoch glaub ich auch, dass viele Webentwickler etwas festgefahren sind und auf ihre gewohnten Tools zurückgreifen werden.

Daher baue ich eher auf die neue Generation der Webentwickler.

---

<sup>8</sup> (Magazin, 2015)