Vorwort

Dieses Buch ist für alle möglichen Programmierer gedacht: professionelle JavaScript-Entwickler, C#-Nutzer, Java-Sympathisanten, Python-Liebhaber, Ruby-Experten, Haskell-Nerds. Egal, welche Sprache Sie bevorzugen – solange Sie etwas Programmiererfahrung haben und sich mit den Grundlagen von Funktionen, Variablen, Klassen und Ausnahmen auskennen, ist dieses Buch für Sie geeignet. Erfahrung mit JavaScript und ein Grundwissen zu Document Objekt Model (DOM) und Netzwerk sind hilfreich. Obwohl wir diese Themen nicht bis ins letzte Detail behandeln, halten sie doch einige ausgezeichnete Beispiele bereit. Wenn Sie mit diesen Konzepten jedoch gar nicht vertraut sind, werden die Beispiele für Sie nicht viel Sinn ergeben.

Unabhängig davon, welche Programmiersprache Sie in der Vergangenheit benutzt haben, teilen wir doch die gleichen Erfahrungen. Das Aufspüren von Ausnahmen und das zeilenweise Durchgehen unseres Codes, um die Probleme zu finden und sie zu beheben, kommen in jeder Sprache vor. TypeScript versucht (recht erfolgreich), Ihnen diese unangenehmen Erfahrungen zu ersparen, indem es Ihren Code automatisch untersucht und Sie auf die übersehenen Fehler hinweist.

Es ist kein Problem, wenn Sie noch nie mit einer statisch typisierten Sprache gearbeitet haben. Ich werden Ihnen den Umgang mit Typen beibringen und zeige Ihnen, wie sie effektiv genutzt werden können, um Programmabstürze zu vermeiden, Ihren Code besser zu dokumentieren, und wie Sie Ihre Applikationen für mehr Benutzer, Entwickler und Server skalieren können. Ich versuche, ausschweifende Erklärungen zu vermeiden und Ideen intuitiv, leicht zu merken und praktisch zu vermitteln. Dabei verwende ich viele Beispiele, um die Dinge möglichst konkret zu halten.

Im Gegensatz zu anderen Sprachen ist TypeScript unglaublich praktisch. Es besitzt vollkommen neue Konzepte, mit denen Sie knapp und präzise formulieren können, was Sie wollen. Das sorgt nicht nur für moderne und sichere Applikationen, sondern auch dafür, dass Sie dabei Spaß haben.

Wie dieses Buch aufgebaut ist

Dieses Buch verfolgt zwei Ziele: Es soll Ihnen ein tiefgehendes Verständnis von der Funktionsweise von TypeScript vermitteln (Theorie) und Ihnen gleichzeitig möglichst viele praktische Ratschläge zum Schreiben von Code für Produktionsumgebungen (Praxis) geben.

Und weil TypeScript eine so praktische Sprache ist, liegen Theorie und Praxis meist sehr nah beieinander. Der Großteil dieses Buchs ist daher eine Mischung aus beidem. Dabei geht es in den ersten Kapiteln hauptsächlich um Theorie, während die andere Hälfte sich fast nur mit der Praxis beschäftigt.

Ich beginne mit den Grundlagen zu Compilern, Typecheckern und Typen. Danach gebe ich Ihnen einen groben Überblick über die verschiedenen Typen und Typoperatoren in TypeScript und ihre Verwendung. Das Gelernte verwenden wir dann, um uns mit fortgeschrittenen Themen zu beschäftigen. Dazu gehören beispielsweise die fortschrittlichsten Merkmale von TypeScripts Typsystem, die Fehlerbehandlung und asynchrone Programmierung. Zum Abschluss zeige ich Ihnen, wie Sie TypeScript mit Ihren Lieblings-Frameworks (Frontend und Backend) verwenden können, wie Sie bestehende JavaScript-Projekte zu TypeScript migrieren können und was Sie bei der Ausführung Ihrer TypeScript-Applikationen in einer Produktionsumgebung beachten müssen.

Die meisten Kapitel enthalten am Ende eine Reihe von Übungen. Versuchen Sie, diese Übungen selbst zu lösen. So bekommen Sie ein tieferes Verständnis der behandelten Inhalte, als es reines Lesen ermöglichen kann. Die Lösungen für die Übungen finden Sie online unter *https://github.com/bcherny/programming-typescript-answers*.

Coding-Stil

Ich habe versucht, über das gesamte Buch einen Coding-Stil beizubehalten. Einige Aspekte dieses Stils folgen eher meinen persönlichen Vorlieben. Zum Beispiel:

- Ich verwende Semikola (;) nur bei Bedarf.
- Für Einrückungen verwende ich grundsätzlich zwei Leerzeichen.
- Handelt es sich bei einem Codebeispiel nur um einen kurzen Abschnitt oder ist die Programmstruktur wichtiger als die Details, verwende ich kurze Variablennamen wie `a`, `f` oder `_`.

Dann gibt es aber noch Aspekte des Coding-Stils, die Sie meiner Meinung nach ebenfalls verwenden sollten. Dies sind unter anderem:

- Verwenden Sie die aktuellste JavaScript-Syntax und Sprachmerkmale (die neueste JavaScript-Version wird üblicherweise als *esnext* bezeichnet). Dadurch hält sich Ihr Code an die neuesten Standards, was die Interoperabilität und die Auffindbarkeit in Suchmaschinen erhöht und Ihnen möglicherweise sogar bei der Akquise neuer Aufträge hilft. Außerdem können Sie dadurch die Vorteile

mächtiger moderner JavaScript-Features wie Pfeil-Funktionen, Promises und Generatoren nutzen.

- Meistens sollten Sie Ihre Datenstrukturen mithilfe des Spread-Operators (`...`) als immutabel kennzeichnen.[1]
- Stellen Sie sicher, dass alles einen Typ besitzt, der nach Möglichkeit automatisch abgeleitet (inferred) werden kann. Verwenden Sie nicht unnötig explizite Typen. So bleibt Ihr Code klar und knapp. Gleichzeitig erhöht sich die Sicherheit, weil inkorrekte Typen klar erkennbar sind, anders als wenn Sie sie nur behelfsmäßig verarzten.
- Halten Sie Ihren Code wiederverwendbar und allgemein. Polymorphismus ist dabei Ihr bester Freund (siehe »Polymorphismus« auf Seite 66).

Diese Ideen sind natürlich nicht neu. Aber TypeScript funktioniert besonders gut, wenn Sie sich danach richten. Der TypeScript-eigene Downlevel-Compiler (TSC), die Unterstützung für schreibgeschützte Typen, eine mächtige Typableitung, Unterstützung für Polymorphismus und ein vollständig strukturelles Typsystem unterstützen einen guten Coding-Stil. Dabei bleibt die Sprache unglaublich ausdrucksstark und dem zugrunde liegenden JavaScript treu.

Noch ein paar weitere Hinweise, bevor wir anfangen:

JavaScript legt keine Pointer und Referenzen offen. Stattdessen verwendet es Werte- und Referenztypen. Werte sind immutabel. Hierzu gehören beispielsweise Strings, Zahlen und boolesche Werte, während Referenzen auf oftmals mutable Datenstrukturen verweisen, wie Arrays, Objekte und Funktionen. Wenn ich in diesem Buch das Wort »Wert« verwende, ist damit in etwa ein JavaScript-Wert oder eine -Referenz gemeint.

Ein paar Worte zum Abschluss: Manchmal muss Ihr Code mit JavaScript oder nicht korrekt typisierten Drittanbieter-Bibliotheken zusammenarbeiten, oder Sie haben es eilig. Das führt nicht unbedingt zu makellosem TypeScript-Code. Dieses Buch zeigt größtenteils, wie Sie TypeScript schreiben *sollten,* und versucht Ihnen klarzumachen, warum Kompromisse möglichst zu vermeiden sind. In der Praxis entscheiden Sie und Ihr Team jedoch selbst, wie Sie vorgehen.

In diesem Buch verwendete Konventionen

Folgende typografische Konventionen werden in diesem Buch verwendet:

Kursiv
: Bezeichnet neue Begriffe, URLs, E-Mail-Adressen, Dateinamen und Dateiendungen.

1 Ein JavaScript-Beispiel: Angenommen, Sie haben ein Objekt `o`, dem die Eigenschaft `k` mit dem Wert `3` hinzugefügt werden soll. Dann können Sie `o` entweder direkt verändern, indem Sie schreiben `o.k = 3`, oder Sie können die Änderungen durch die Erstellung eines *neuen* Objekts durchführen: `let p = {...o, k: 3}`.

`Nichtproportionalschrift`
: Wird verwendet für Codebeispiele sowie innerhalb von Absätzen für Programmelemente wie Variablen- oder Funktionsnamen, Datentypen, Umgebungsvariablen, Anweisungen und Schlüsselwörter.

`Nichtproportionalschrift kursiv`
: Steht für Text, der durch Benutzereingaben ersetzt wird, oder Werte, die durch den Kontext bestimmt werden.

Dieses Element kennzeichnet einen Tipp oder Vorschlag.

Dieses Element bezeichnet einen allgemeinen Hinweis.

Dieses Element kennzeichnet eine Warnung oder einen Gefahrenhinweis.

Codebeispiele in diesem Buch

Zusätzliches Material (Codebeispiele, Übungen etc.) können Sie unter folgender Adresse herunterladen: *https://github.com/bcherny/programming-typescript-answers*.

Dieses Buch soll Ihnen bei Ihrer Arbeit helfen. Grundsätzlich dürfen Sie den Beispielcode aus diesem Buch in Ihren Programmen und Ihrer Dokumentation nutzen. Sie müssen hierfür keine ausdrückliche Genehmigung einholen, es sei denn, es handelt sich um eine größere Menge Code. Wenn Sie beispielsweise ein Programm schreiben, das mehrere Codeabschnitte aus diesem Buch verwendet, ist keine Erlaubnis nötig; für den Verkauf oder Vertrieb einer CD-ROM mit Beispielen aus O'Reilly-Büchern dagegen schon. Das Beantworten einer Frage durch Zitieren von Beispielcode erfordert keine Erlaubnis. Verwenden Sie einen erheblichen Teil der Beispielcodes aus diesem Buch in Ihrer Dokumentation, ist jedoch unsere Erlaubnis nötig.

Eine Quellenangabe ist zwar erwünscht, aber nicht unbedingt notwendig. Hierzu gehört in der Regel die Erwähnung von Titel, Autor, Verlag und ISBN, zum Beispiel: »*Programming TypeScript* von Boris Cherny (O'Reilly). Copyright 2019 Boris Cherny, 978-1-492-03765-1.«

Sollten Sie nicht sicher sein, ob die Nutzung der Codebeispiele außerhalb der hier erteilten Genehmigung liegt, nehmen Sie bitte unter der Adresse *permissions@oreilly.com* Kontakt mit uns auf.

Website zum Buch

Zu diesem Buch gibt es eine Website, auf der Sie Errata, Beispiele und weitere Informationen finden können. Sie finden die Website unter: *https://oreil.ly/programming-typescript*.

Danksagungen

Dieses Buch ist das Ergebnis aus Jahren des Sammelns von Schnipseln und Skizzen, gefolgt von Jahren des Schreibens – frühmorgens, an Wochenenden und bei Nacht.

Vielen Dank an O'Reilly für die Möglichkeit, dieses Buch zu schreiben, und an meine Lektorin Angela Rufino für die Unterstützung während des gesamten Prozesses. Dankeschön auch an Nick Nance für seinen Beitrag über »Typsichere APIs« auf Seite 212 und an Shyam Seshadri für seinen Beitrag zu »Angular 6/7« auf Seite 208. Ein Dankeschön an meine technischen Sachverständigen. Das sind: Daniel Rosenwasser vom TypeScript-Team, der *sehr* viel Zeit damit verbracht hat, dieses Manuskript zu lesen und mich durch die Feinheiten von TypeScripts Typsystem zu leiten. Ein weiteres Dankeschön an Jonathan Creamer, Yakov Fain, Paul Buying und Rachel Head für technische Anpassungen und ihr Feedback. Mein herzlicher Dank gilt auch meiner Familie, Liza und Ilya, Vadim, Roza und Alik, Faina und Yosif, die mich ermutigt haben, dieses Projekt durchzuführen.

Meine größte Dankbarkeit gilt meiner Partnerin Sara Gilford, die mich während des Schreibens unermüdlich unterstützt hat, selbst wenn dadurch unsere Wochenendpläne durchkreuzt wurden oder es spätnächtliches Schreiben und Programmieren und viel zu viele unerwartete Gespräche über die Details von Typsystemen bedeutete. Ohne dich hätte ich das nicht geschafft. Ich werde dir immer dankbar für deine Unterstützung sein.