Deutsche Ausgabe

Flutter & Dart Kochbuch

Cross-Platform-Apps für die Cloud entwickeln

Richard Rose

Übersetzung von Thomas Demmig

Flutter & Dart Kochbuch

Cross-Platform-Apps für die Cloud entwickeln

Richard Rose

Deutsche Übersetzung von Thomas Demmig

O'REILLY®

Richard Rose

Übersetzung: Thomas Demmig
Lektorat: Sandra Bollenbacher
Fachlektorat: Thomas Künneth
Copy-Editing: Petra Heubach-Erdmann, Düsseldorf
Satz: inpunkt[w]o, Wilnsdorf, *www.inpunktwo.de*
Herstellung: Stefanie Weidner, Frank Heidt
Druckerei: mediaprint solutions GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

ISBN:
Print 978-3-96009-229-2
PDF 978-3-96010-800-9
ePub 978-3-96010-801-6
mobi 978-3-96010-802-3

1. Auflage 2024

Wieblinger Weg 17
69123 Heidelberg

Authorized German translation of the English edition of *Flutter and Dart Cookbook*
ISBN 9781098119515 © 2023 Richard Rose

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«.
O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

Hinweis:
Dieses Buch wurde mit mineralölfreien Farben auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie. Hergestellt in Deutschland.

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: *kommentar@oreilly.de*.

5 4 3 2 1 0

Inhalt

Vorwort

Jeden Tag treffe ich auf Leute, die mit Flutter und Dart tolle Apps bauen und großartige Inhalte schaffen. Sowohl Flutter als auch Dart helfen Entwicklerinnen und Entwicklern dabei, schneller zu lernen und Träume zu verwirklichen. Als Rich Rose mir erzählt hat, dass er gerade dieses Buch schreibt, wusste ich, dass er damit Tausenden Menschen helfen würde.

Der praxisorientierte und auf Beispielen aufbauende Ansatz dieses Kochbuchs bietet eine faszinierende Möglichkeit, Flutter und Dart zu erlernen. Als frühzeitiger Anwender von Flutter und Dart kann Rich unfassbar viele wertvolle Tipps und Tricks teilen. Es ist eine großartige Referenz, in der Sie Antworten auf viele Fragen finden, denen Sie sich in Ihrer täglichen Arbeit eventuell gegenübersehen.

Ich freue mich darüber, dieses Buch empfehlen zu können, und ich hoffe, es hilft Ihnen dabei, das Potenzial der fantastischen Erfahrung beim Bauen von Flutter-Apps und bei der Arbeit mit der Programmiersprache Dart ausschöpfen zu können.

– Majid Hajian,
Google Developer Expert für Flutter und Dart und
Head of Developer Relations bei Invertase
Oslo, November 2022

Einleitung

Willkommen zum *Flutter & Dart Kochbuch*. Falls Sie noch nichts von Flutter gehört haben: Es handelt sich um das Multiplattform-Framework, das die Entwicklungs-Community im Sturm erobert. Dart stellt ein umfassendes Software Development Kit (SDK) bereit, das die Grundlage von Flutter liefert. Statt mehrere Technologien erlernen zu müssen, ermöglicht Flutter es Ihnen, Android, iOS, Linux, das Web und Windows über eine einzige Codebasis zu bedienen.

Als jemand, der viel auf YouTube unterwegs ist, bin ich immer von den erstellten Beispielanwendungen beeindruckt. Flutter und Dart haben nicht nur dafür gesorgt, dass ich wieder Spaß am Programmieren gefunden habe, sondern auch, dass ich ein paar coole Leute aus der Flutter-Community treffen konnte.

Die Flutter-Community ist ziemlich gut drauf und stellt qualitativ hochwertigen Content bereit – Grüße gehen raus an die YouTuber- und Google-Developer-Communities, die Zeit und Aufwand investieren, um anderen dabei zu helfen, einen Fuß in die Softwarebranche zu kriegen.

Egal, ob Sie viel Erfahrung in der Entwicklung haben oder gerade erst beginnen – Flutter sorgt dafür, dass Entwickeln Spaß macht. Es lässt sich sehr schnell erlernen, ist ausgesprochen mächtig und erlaubt die Integration mit leistungsfähigen cloudbasierten Lösungen wie Firebase. Fangen Sie jetzt an und bauen Sie eine Anwendung für die nächste eine Million Benutzerinnen und Benutzer.

Wer dieses Buch lesen sollte

Egal, ob Sie Flutter und Dart kennen oder nicht – Sie haben den unwiderstehlichen Drang, etwas zu bauen. Ignorieren Sie den Wunsch zur Prokrastination und schaffen Sie etwas für die Zukunft. Wie wird es Ihnen möglich sein, motiviert zu bleiben und so weit zu kommen, dass Sie eine Anwendung ausliefern können? Lesen Sie dieses Buch. Das Schöne beim Lernen von Flutter und Dart ist, dass es eine sehr niedrige Einstiegshürde hat.

Zum Entwickeln einer Anwendung muss man Fertigkeiten besitzen und Leistung zeigen. Es gibt einen Grund dafür, dass Softwareentwickler und -entwicklerinnen so viel Geld verdienen. Egal, ob Sie mehrere Programmiersprachen beherrschen oder noch gar keine Erfahrung beim Entwickeln von Software besitzen – Flutter und Dart sind ein großartiger Ausgangspunkt für das Schaffen erstaunlicher Anwendungen. Sie können vom ersten Tag an produktiv werden und mit wenig Aufwand schöne Applikationen erstellen.

Beim Schreiben von Anwendungen sind viele Bestandteile zu berücksichtigen und auch externe Services können dazugehören. Lernen Sie die Grundlagen der Arbeit mit der Firebase-Suite kennen und machen Sie Ihre ersten Schritte in der Cloud. Steigen Sie in das Programmieren von Spielen ein und erfahren Sie, wie Sie Blöcke für die Flame-Spiele-Engine schreiben.

Warum ich dieses Buch geschrieben habe

Auf Flutter stieß ich das erste Mal, als meine Frau anfing, die Sprache mithilfe eines Flutter-Entwicklungskurses von Google zu erlernen. Besonders gefiel mir die Schnelligkeit, mit der Anwendungen geschaffen wurden, und wie schön sie aussahen. Der direkte Produktivitätsschub war beeindruckend und ich fand mich schnell im Bauen einfacher Multiplattform-Apps wieder.

Als ich damit begann, diese neue Technologie zu erlernen, gab es eine Reihe von Dingen, die entweder nicht offensichtlich waren oder die man sich schlecht merken konnte. Ich wäre sehr froh gewesen, wenn dieses Kochbuch schon neben mir auf dem Schreibtisch gelegen hätte, als ich mich durch diesen Kurs gearbeitet habe. Jetzt stehen noch viel mehr ausgezeichnete Online-Kurse zur Verfügung, aber die Frage bleibt: Wie erledigen Sie *x* in Flutter? Manchen wird dieses Buch eine Ergänzung zu ihrem bestehenden Wissen sein. Anderen mag es als Sicherheitsnetz dienen, das sie auf ihrer Reise in der Flutter-Entwicklung leitet und unterstützt.

Wie dieses Buch aufgebaut ist

Bei jedem Neueinstieg in eine Sprache gibt es die Frage, wie Sie am besten loslegen und was Sie wissen müssen. Ich empfehle Ihnen durchaus, dieses Buch von vorne nach hinten durchzuarbeiten, aber realistischerweise werden Sie direkt in die Flutter-Kapitel springen wollen. Daher enthält dieses Buch größere Themen, in denen Sie nach Bedarf in Fragestellungen einsteigen können. Das sind Folgende:

- In den Kapiteln 1 und 2 gebe ich einen Überblick über die Sprache Dart, um Ihnen bei den Grundlagen wie Variablen und Programmablauf zu helfen.
- Die Kapitel 3 bis 6 gehen dann auf die weiteren wichtigsten Elemente von Dart ein, die dafür sorgen, dass Sie die Sprache produktiv nutzen können.

- In den Kapiteln 7 bis 14 geht es um die Sprache Flutter und Sie erfahren die Grundlagen des Renderns von Widgets auf dem Bildschirm. Ich vermute, das sind die Kapitel, auf die Sie am häufigsten zurückgreifen werden, wenn Sie Anwendungen entwickeln, bei denen Widgets erstellt und Daten verwaltet werden müssen.
- Die Kapitel 15 und 16 drehen sich um das Arbeiten in der Cloud und spezifischer mit Firebase, um Authentifizierung, Datenbanken und Hosting ergänzen zu können.
- Kapitel 17 kümmert sich um den Einsatz der Spiele-Engine Flame und Sie erhalten einen Überblick über das Paket. Haben Sie ein Wochenende Zeit, können Sie damit das Spiel Frogger nachbauen.
- Im Anhang »Richten Sie Ihre Umgebung ein« erfahren Sie, wie Sie Ihre Umgebung aufsetzen – die Installation des Flutter-Frameworks, der Einsatz von Flutter Doctor und das Arbeiten in einer IDE.

Konventionen, die in diesem Buch genutzt werden

Die folgenden typografischen Konventionen kommen in diesem Buch zum Einsatz:

Kursiv
: Steht für neue Begriffe, URLs, E-Mail-Adressen, Dateinamen und Dateierweiterungen.

`Schreibmaschinenschrift`
: Steht für Programmlistings, aber auch innerhalb von Absätzen, um sich auf Programmelemente wie Variablen oder Funktionsnamen, Datenbanken, Datentypen, Umgebungsvariablen, Anweisungen und Schlüsselwörter zu beziehen.

`fette Schreibmaschinenschrift`
: Steht für Befehle oder anderen Text, der genau so einzugeben ist.

Dieses Element enthält einen allgemeinen Hinweis.

Dieses Element enthält einen Tipp oder Vorschlag.

Danksagung

Dieses Buch ist meiner geliebten Frau gewidmet, die mich dazu inspiriert hat, Flutter zu lernen – damit ich die Fehler in ihren Anwendungen beheben kann :-). Dank geht an meine Familie, die immer Zeit hatte, laut Musik laufen zu lassen oder auf ein spontanes Schwätzchen vorbeizukommen, obwohl Dad doch gerade hochkonzentriert an einem Buch schrieb. Aber im Ernst: vielen Dank an Dawn, Bailey, Elliot, Noah und Amelia. Im Laufe eines Jahres wurde das Arbeiten an diesem Buch zu mehr als einem Projekt – es wurde zu einer Inspiration.

Besonderer Dank geht an Dylan Peck, Casey Palowitch, Allesandro Palmieri und Andre Brogdon, die mich unterstützten und für die Möglichkeit sorgten, Flutter in der Google Developer Group Community vorzustellen. Genau so etwas scheint den Unterschied auszumachen und aus persönlicher Erfahrung kann ich sagen, dass diese Sessions gerne angenommen wurden.

Ich möchte mich auch bei den technischen Korrektoren – Alex Moore, Rob Edwards und Majid Hajian – für ihre Vorschläge und Rückmeldungen bedanken. Es ist zwar nicht leicht, ein Buch zu schreiben, aber es hilft ungemein, Leute fragen zu können, die ihre knappe Zeit gerne opfern und dabei helfen, so etwas möglich zu machen. Ich weiß die Zeit und den Aufwand wirklich zu schätzen, die jeder von euch investiert hat.

Ein großer Dank geht auch an Jeff Bleiel, der ein fantastischer Lektor ist und den ganzen Prozess viel angenehmer und weniger anstrengend gemacht hat, als er hätte sein können. Vielen Dank auch an Zan McQuade und Jonathon Owen von O'Reilly.

KAPITEL 1

Dart-Variablen kennenlernen

In diesem Kapitel geht es darum, die Grundlagen des Variableneinsatzes in Dart kennenzulernen. Wie Sie vermutlich schon erwarten, bietet Dart eine ganze Menge Variablentypen an. Um mit der Sprache schnell vertraut zu werden, ist es ausgesprochen wichtig, die grundlegenden Datentypen zu kennen.

Sind Sie mit dem Einsatz von Variablen in anderen Programmiersprachen vertraut, sollte es nicht allzu schwierig sein, das auch in Dart zu verstehen. Nutzen Sie dieses Kapitel als Schnelleinstieg, um Ihr Verständnis zu festigen, bevor Sie sich den komplexeren Aufgaben zuwenden.

Einsteigern und Einsteigerinnen wird dieses Kapitel die Grundlagen vermitteln. Es soll auch als Begleitung dienen, wenn Sie mit Ihrer Reise beim Erlenen von Dart und Flutter fortfahren.

Im Kapitel sind die Codebeispiele in sich abgeschlossen und sie konzentrieren sich jeweils auf einen typischen Anwendungsfall. Ich beginne damit, die vier wichtigsten Variablentypen vorzustellen (`int`, `double`, `bool` und `String`) und zu zeigen, wie sie eingesetzt werden. Dann werden Sie erfahren, wie Sie Dart wissen lassen, was Sie mit Ihren Variablen tun wollen (`final`, `const` und `null`).

Sie werden sich auch mit Unveränderbarkeit befassen – wo es darum geht, ob sich der einer Variablen zugewiesene Wert verändert werden kann. In Dart machen die Schlüsselwörter `const` und `final` eine Variable unveränderbar. Es gibt einen kleinen Unterschied, ob die Variable zur Kompilierungszeit oder zur Laufzeit geprüft wird. Zur Kompilierungszeit bedeutet, dass Prüfungen auf dem Code beim Bauen angewendet werden (für `const`-Variablen), während Prüfungen zur Laufzeit beim Ausführen der Anwendungen durchgeführt werden (für `final`-Variablen).

Seit Dart 2.0 ist die Sprache typsicher – sobald eine Variable deklariert wurde, kann der Typ nicht mehr geändert werden. Wird beispielsweise eine Variable vom Typ `double` deklariert, kann sie nicht als `int` genutzt werden, ohne sie explizit zu casten. Bevor ich genauer auf das Verwenden von Variablen eingehe, muss die Programmierumgebung sauber aufgesetzt sein.

Eine Dart-Anwendung ausführen

Problem

Sie wollen eine Dart-Anwendung ausführen.

Lösung

Dart-Code lässt sich in Ihrer Umgebung ausführen, sobald das SDK installiert wurde. Um zu lernen, wie Sie das SDK installieren, schauen Sie in »Richten Sie Ihre Umgebung ein«. Öffnen Sie ein Terminal, um Befehle eingeben zu können. Nutzen Sie eine IDE, muss das Terminal innerhalb dieser Anwendung geöffnet werden. Prüfen Sie nun, ob Dart auf dem Gerät installiert ist, indem Sie die Version wie folgt ausgeben lassen:

```
dart --version
```

Konnte der Befehl erfolgreich ausgeführt werden, sollten Sie die Version des installierten SDK und die Plattform sehen, auf der es läuft. Gab es ein Problem, werden Sie die Installation und den Pfad auf Ihrem Rechner prüfen müssen.

Erstellen Sie jetzt in Ihrem Editor eine neue Datei mit dem Namen *main.dart* und geben Sie folgenden Code ein:

```
void main() {
  print('Hallo Dart-Welt!');
}
```

Führen Sie Ihren Beispielcode an der Befehlszeile wie folgt aus:

```
dart main.dart
```

Dieser Befehl sollte »Hallo Dart-Welt!« ausgeben.

Diskussion

Der `dart`-Befehl steht als Teil der Dart-SDK-Installation zur Verfügung. In diesem Beispiel führt er eine Datei namens *main.dart* aus. Dart-Anwendungen haben die Erweiterung *.dart* und sie können entweder an der Befehlszeile oder innerhalb einer IDE (zum Beispiel Android Studio oder VS Code) laufen. Beachten Sie, dass weder Android Studio noch VS Code standardmäßig dafür eingerichtet sind, Dart/Flutter nutzen zu können. Sie werden (neben dem Installieren des SDK) die entsprechenden Plugins installieren müssen, bevor Sie Code ausführen können.

Wollen Sie Dart nicht in Ihrer Umgebung installieren, nutzen Sie den Online-Editor DartPad, der unter *https://dartpad.dev* zur Verfügung steht.

Können Sie den Befehl `dart` nicht ausführen, ist es sehr wahrscheinlich, dass das SDK nicht korrekt installiert wurde. Nutzen Sie die neuesten Installationsanweisungen (*https://dart.dev/get-dart*), um die Installation auf Ihrem Rechner sauber umzusetzen.

Mit Integerwerten arbeiten

Problem

Sie wollen eine Zahl ohne Dezimalkomma speichern.

Lösung

Nutzen Sie eine Integervariable, um eine Zahl ohne Dezimalkomma zu speichern.

Wollen Sie den ganzzahligen Wert 35 ablegen, sähe die Deklaration wie folgt aus:

```
void main() {
  int    myVariable = 35;

  print(myVariable);
}
```

In der Programmiersprache Dart nutzt ein Integerwert die Referenz `int`. In diesem Codebeispiel ist der Datentyp (zum Beispiel `int`) der erste Teil der Deklaration. Danach wird der Datentyp mit einem Label verbunden (hier `myVariable`). Schließlich weisen Sie dem verbundenen Label noch einen Wert zu – in diesem Beispiel den Wert 35.

Um den Datentyp zu verwenden, wird eine Variable deklariert, zum Beispiel `myVariable`. Eine Variable ist ein Label, das genutzt wird, um den erstellten Datentyp zu referenzieren.

Steht eine Variable einmal zur Verfügung, können Sie dem Datentyp einen Wert zuweisen. Im Beispiel wird die Ganzzahl `35` der Variablen `myVariable` zugewiesen. Eine `print`-Anweisung gibt dann den Wert der Variablen aus.

Diskussion

Dart nutzt eine Reihe von Patterns, um Variablen zu deklarieren. Das Präfix deklariert den zu verwendenden Datentyp, dann folgt ein Variablenlabel und dann optional eine Zuweisung.

In diesem Beispiel ist der verwendete Datentyp ein `int`. Ein Integer repräsentiert Zahlen, die keinen Dezimalpunkt haben. Der typische Anwendungsfall für ein `int` ist eine Zahl, die keinen Dezimalpunkt erfordert (zum Beispiel aus Gründen der Präzision). Ein Integer ist als 64-Bit-Ganzzahl definiert. Der Integer-Datentyp ist ein Subtyp von `num`, der grundlegende Operationen enthält, zum Beispiel +/-.

Das Variablenlabel dient als Möglichkeit, auf den `int`-Datentyp im Beispielcode zuzugreifen. Dieses Beispiel nutzt das Label `myVariable`. Versuchen Sie, beim Benennen einer Variablen, ihren Zweck im Namen widerspiegeln zu lassen.

Um die Variablendeklaration zu vervollständigen, wird ein Wert zugewiesen. Hier weisen Sie Ihrer Variablen den Wert `35` zu. Referenzieren Sie diese Variable, erwarten Sie, dass für den `int`-Datentyp der Wert 35 genutzt wird.

Wäre die Integervariable nicht initialisiert worden, könnte auf den Wert nicht zugegriffen werden. In diesem Fall gibt Dart eine Fehlermeldung zurück, die darauf hinweist, dass der Wert, auf den Sie zugreifen wollen, eine nicht-nullbare Variable ist. Das heißt im Prinzip, dass Sie einer Variablen, auf die Sie zugreifen wollen, keinen Wert zugewiesen haben. In den meisten Fällen ist das ein Fehler und der Dart-Compiler wird freundlicherweise darauf hinweisen, dass Sie einen Fehler gemacht haben. Wollten Sie tatsächlich eine nullbare Variable verwenden, müssten Sie Dart mitteilen, wie es mit dieser Situation umgehen soll. Mehr dazu erfahren Sie in Abschnitt »Mit Nullvariablen arbeiten«.

Mit Doublewerten arbeiten

Problem

Sie wollen eine Zahl mit einem Dezimalkomma speichern.

Lösung

Nutzen Sie eine Double-Variable (also eine Variable mit doppelter Präzision), um eine Zahl mit einem Dezimalkomma zu speichern.

Wollen Sie einen Doublewert von 2,99 speichern, deklarieren Sie das Folgende:[1]

```
void main() {
  double myVariable = 2.99;

  print(myVariable);
}
```

So wie bei anderen Variablen steht hier zuerst der gewünschte Datentyp, zum Beispiel `double`. Dann ist ein Label für die Variable erforderlich, zum Beispiel `myVariable`. Schließlich weisen Sie noch einen Wert zu – in diesem Beispiel den Wert 2,99.

Diskussion

In diesem Beispiel geben Sie zunächst den zu verwendenden Datentyp an, also `double`. Danach folgt ein Variablenname (hier `myVariable`) für `double`. Der letzte Teil, bei dem Sie der Variablen einen Wert zuweisen, ist optional.

Der typische Anwendungsfall für einen `double`-Datentyp ist eine Zahl, die eine gewisse Genauigkeit erfordert. Ein `double`-Datentyp ist eine 64-Bit-Gleitkommazahl. Double ist ein Subtyp von `num`, der grundlegende Operationen wie +/- enthält.

1 In Dart (und in so gut wie allen anderen Programmiersprachen) muss statt des Dezimalkommas ein Dezimalpunkt verwendet werden. 2,99 wird dort daher als 2.99 geschrieben.

Mit booleschen Werten arbeiten

Problem

Sie wollen einen Wahr/Falsch-Wert ablegen.

Lösung

Nutzen Sie eine `bool`-Variable, um einen Wahr/Falsch-Status zu speichern.

Deklarieren Sie eine boolesche Variable mithilfe des Schlüsselworts `bool`, gefolgt von einem Label für den Variablennamen, zum Beispiel `myVariable`. Weisen Sie der Variablen schließlich einen Wert `true` oder `false` zu.

Hier ein Beispiel für das Deklarieren einer `bool`-Variablen:

```
void main() {
  bool myVariable = true;

  print(myVariable);
}
```

Diskussion

In diesem Beispiel haben Sie zuerst den zu verwendenden Datentyp angegeben, hier `bool`. Danach folgt ein Variablenname für den definierten Datentyp. Der letzte Teil ist optional – hier weisen Sie der benannten Variablen einen Wert zu.

Der Anwendungsfall für ein `bool` ist ein Wahr/Falsch-Szenario. Beachten Sie, dass `true` und `false` in Dart reservierte Schlüsselwörter sind. *Reserviert* heißt hier, dass das Wort in der Sprache eine definierte Bedeutung besitzt. Ein boolescher Datentyp enthält logische Operationen, wie zum Beispiel Und, Gleichheit sowie inklusives und exklusives Oder.

Mit Strings arbeiten

Problem

Sie wollen eine Zeichenfolge speichern.

Lösung

Nutzen Sie eine `String`-Variable, um eine Zeichenfolge abzulegen.

Hier ein Beispiel für das Deklarieren eines `String`s:

```
void main() {

  String myVariable  = "Ich bin ein String";

  String myVariable2 = """
    Ich bin ein mehrzeiliger
    String
    """;

  print(myVariable);
  print(myVariable2);
}
```

Diskussion

Zuerst geben Sie den zu verwendenden Datentyp an, hier `String`. Ein String dient dazu, eine Folge von Zeichen zu repräsentieren, bei denen es sich um Ziffern und Buchstaben handeln kann. Beachten Sie, dass ein String mit einem großen »S« beginnt, was eine beliebte Fehlerquelle ist, wenn man Dart erlernt.

Der Datentyp erfordert eine Variable für den definierten Datentyp und diese wird genutzt, um den zugewiesenen Wert zu referenzieren. Eine Zuweisung eines Werts ist an dieser Stelle optional.

Der typische Anwendungsfall für einen `String`-Datentyp ist eine Zeichenfolge. Ein `String`-Datentyp in Dart nutzt Code-Units aus dem 16-Bit Unicode Transformation Format (UTF-16). Die Klasse `String` dient dazu, Textzeichen zu repräsentieren, aber durch die Codierung können auch andere Zeichen genutzt werden, zum Beispiel Emojis.

Beim Einsatz einer `String`-Variablen können Sie entweder zusammenpassende einfache oder doppelte Anführungszeichen verwenden, um den anzuzeigenden Text festzulegen. Brauchen Sie mehrzeiligen Text, können Sie das mit dreifachen Anführungszeichen erreichen. Im Beispiel sehen Sie beide Arten von Deklarationen.

Informationen an der Konsole ausgeben

Problem

Sie wollen per Programm Informationen aus einer Dart-Anwendung ausgeben.

Lösung

Nutzen Sie eine `print`-Anweisung, um Informationen aus einer Anwendung auszugeben. Die `print`-Anweisung kann sowohl statische (also ein Stringliteral) als auch variable Inhalte ausgeben.

Hier ein Beispiel für das Ausgeben statischen Inhalts:

```
void main() {

  print('Hallo Welt!');
}
```

Und hier eines, bei dem der Inhalt von Variablen ausgegeben wird:

```
void main() {

  int intVariable  = 10;
  var boolVariable = true;

  print(intVariable);
  print('$intVariable');
  print('Die boolesche Variable ist $boolVariable');
}
```

Diskussion

Mit dem `$`-Zeichen können Sie eine Variable in einer `print`-Anweisung referenzieren. Setzen Sie vor eine Variable ein `$`, teilen Sie Dart mit, dass eine Variable genutzt wird und dieser Wert eingesetzt werden soll.

Die `print`-Anweisung ist in einer Reihe von Szenarien nützlich. Für das Ausgeben statischer Inhalte sind keine zusätzlichen Schritte erforderlich, um Informationen anzuzeigen. Umschließen Sie den Wert in Anführungszeichen, kümmert sich die `print`-Anweisung um den Rest.

Um den Wert einer Variablen auszugeben, müssen Sie vor ihren Namen ein `$`-Zeichen stellen. Dann tauscht Dart den Namen durch den Wert der Variablen aus. Im zweiten Beispiel sehen Sie drei gebräuchliche Vorgehensweisen, um eine Variable in einer `print`-Anweisung zu referenzieren.

Dart wird Ihnen mitteilen, ob es geschweifte Klammern benötigt – normalerweise ist das nicht erforderlich. Aber wenn Sie einen komplexen Variablentyp erstellen, müssen Sie darauf achten, dass Sie das gewünschte Element referenzieren.

Eine konstante Variable hinzufügen (Kompilierungszeit)

Problem

Sie wollen eine Variable erstellen, die niemals geändert werden kann (also unveränderbar ist).

Lösung

Nutzen Sie `const`, um eine Variable zu erstellen, deren Wert nicht erneut zugewiesen werden kann und die beim Kompilieren überprüft wird.

Hier ein Beispiel für das Verwenden einer `const`-Variablen:

```
void main() {
  const daysInYear = 365;

  print ('Es gibt $daysInYear Tage im Jahr');
}
```

Diskussion

In Dart steht `const` für einen Wert, der nicht geändert werden kann.

Nutzen Sie das Schlüsselwort `const`, wenn sich eine Variable nicht ändern soll. Im Beispiel wird eine `const`-Variable deklariert, die auf den Wert 365 gesetzt ist – sie ist im Rahmen der Anwendung nicht veränderbar.

Versuchen Sie, den Wert innerhalb der Anwendung zu verändern, erhalten Sie einen Fehler beim Kompilieren, der darauf hinweist, dass eine Zuweisung nicht vorgenommen werden kann, weil die angegebene Variable ein `const` ist. Fügen Sie noch eine Zeile `daysInYear = 10` hinzu, werden Sie diesen Fehler erhalten.

Der Einsatz von `const` ist eine gute Möglichkeit, Fehler in Ihrer Anwendung zu vermeiden. Durch das Deklarieren von Variablen als `const` wird für eine robuste Schnittstelle gesorgt, die den Compiler nutzt, um den Einsatz von Variablen explizit zu überprüfen.

Eine konstante Variable hinzufügen (Laufzeit)

Problem

Sie wollen eine Variable erstellen, die nicht verändert werden kann, aber Sie kennen ihren Wert erst, wenn die Anwendung läuft (also zur Laufzeit).

Lösung

Nutzen Sie `final`, um eine Variable zu erstellen, deren Wert nicht erneut zugewiesen werden kann. Im Gegensatz zu einer `const`-Variablen wird der Wert einer `final`-Variablen zur Laufzeit zugewiesen.

Hier ein Beispiel für den Einsatz einer `final`-Variablen:

```
void main() {
  final today = DateTime.now();

  print('Heute ist ${today.weekday}');
}
```

Diskussion

Das Schlüsselwort `final` kommt in Situationen zum Einsatz, bei denen ein Wert zur Laufzeit ermittelt wird (also wenn die Anwendung aktiv ist), sich danach aber nicht mehr ändert. Auch dieser zugewiesene Wert ist unveränderbar, aber anders als ein `const`-Wert ist er beim Kompilieren noch nicht bekannt.

Versuchen Sie, einer `final`-Variablen einen Wert zuzuweisen, die schon einen Wert zugewiesen hat, wird der Compiler einen Fehler melden.

Im Codebeispiel wird der durch die `print`-Anweisung ausgegebene Tag von der `DateTime`-Funktion zurückgegeben. Dieser wird erst ermittelt, wenn die Anwendung läuft, daher wird er tatsächlich den aktuellen Wochentag liefern.

Mit Nullvariablen arbeiten

Problem

Sie wollen einer Variablen einen Standardwert von null zuweisen.

Lösung

Nutzen Sie `null`, um einer deklarierten Variablen einen konsistenten Wert zuzuweisen. Null ist ein interessantes Konzept, das für das Fehlen eines Inhalts steht. Typischerweise wird ein Nullwert genutzt, um Variablen zu initialisieren, denen kein Standardwert zugewiesen wurde. In diesem Fall kann `null` dazu dienen, eine Variable zu repräsentieren, der nicht explizit ein Wert zugewiesen wurde.

Hier ein Beispiel für das Deklarieren einer Variablen in Dart als null:

```
void main(){
  int ?myVariable;
  print ('Zehn: $myVariable');

  myVariable = 10;
  print ('Zehn: $myVariable');
}
```

Diskussion

Damit einem Datentyp `null` zugewiesen werden kann, wird erwartet, dass der `?`-Typ angefügt wird, um explizit zu zeigen, dass ein Wert auch null sein kann. Im Beispiel wird `myVariable` auf nullbar gesetzt, indem vor dem Variablennamen ein `?` steht.

In Dart ist `null` außerdem ein Objekt, es kann also über den einfachen »kein Wert«-Fall hinaus eingesetzt werden. Aktuelle Versionen des Dart-SDK erfordern eine explizite Angabe, ob ein Datentyp nullbar oder nicht-nullbar ist.

Ab Dart v2.0 ist `null`-Typsicherheit nun Standard, es ist also nicht mehr länger möglich, allen Datentypen `null` zuweisen zu können.

Weitere Informationen dazu erhalten Sie in der Referenz zur Null-Klasse in der Dart-API (*https://oreil.ly/5TfdW*).

KAPITEL 2

Den Programmablauf steuern

Das Steuern des Programmablaufs bedeutet, zu beeinflussen, wie Anweisungen in einer Anwendung ausgeführt werden. Es gibt klassische Ablaufsteuerungen, wie zum Beispiel bedingte Anweisungen und Schleifen, die die Reihenfolge der Anweisungen festlegen. Dart bietet eine Reihe von Methoden an, um den Ablauf der Anwendung abhängig von Entscheidungen zu steuern.

Haben Sie schon andere Sprachen wie Python, JavaScript und so weiter genutzt, werden Sie mit den Inhalten aus diesem Kapitel sehr vertraut sein. Wer neu in der Entwicklung ist, wird dieses Kapitel dringend benötigen! Anweisungen zum Steuern des Programmablaufs sind in den meisten Sprachen, mit denen man zu tun hat, sehr ähnlich. Ein Teil des Lernens einer Sprache dreht sich darum, sich mit dieser Art von Anweisungen vertraut zu machen.

In diesem Kapitel werden Sie lernen, wie Sie den Programmablauf steuern, um Logik in Ihre Anwendung zu bringen. Sie werden zudem Anwendungsfälle für jede Anweisung kennenlernen. Viele der Abläufe nutzen eine bedingte Anweisung, die dazu dient, vorzugeben, welche Aktionen auszuführen sind. Achten Sie besonders auf solche Bedingungen und versuchen Sie, den Programmablauf in Ihrer Anwendung effizient zu steuern.

Überprüfen, ob eine Bedingung erfüllt wurde

Problem

Sie wollen eine logische Prüfung für eine Bedingung durchführen, bevor Sie eine Anweisung ausführen.

Lösung

Nutzen Sie eine `if`-Anweisung, um eine Ablaufsteuerung für eine binäre Option zu erhalten. Mit einer `if`-Anweisung können Sie prüfen, ob eine logische Aussage gültig ist.

Gibt es mehrere Optionen, sollten Sie über den Einsatz einer `switch`-Anweisung nachdenken (siehe Abschnitt »Aktionen abhängig von einem Wert ausführen«).

Dieses Beispiel zeigt, wie Sie die `if`-Bedingung nutzen. Sie dient dazu, den Wert einer `bool`-Variablen zu prüfen. Ist diese auf `true` gesetzt, wird die erste Nachricht ausgegeben. Ist die `bool`-Variable auf `false` gesetzt, wird die Alternative ausgegeben.

```
void main() {

  bool isFootball = true;

  if (isFootball) {
    print('Football ist toll!');
  } else {
    print('Sport ist toll!');
  }
}
```

Diskussion

Mit einer `if`-Anweisung erlangen Sie die Kontrolle über den logischen Ablauf in einer Anwendung. Solch eine Steuerung ist für das Bauen von Anwendungen essenziell und Sie erhalten so einen einfachen Mechanismus, um bei Entscheidungen auswählen zu können. Eine bedingte `if`-Anweisung ist in Abbildung 2-1 zu sehen.

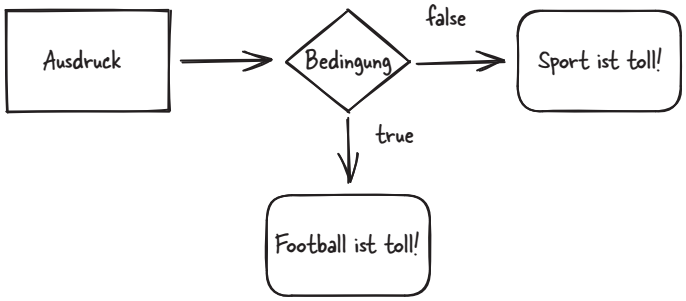

Abbildung 2-1: Ablauf einer `if`*-Anweisung*

In diesem Beispiel ist die Überprüfung bei der `if`-Anweisung implizit – sie prüft, ob der zugewiesene Wert `true` ist. Logische Operatoren wie && (AND) und || (OR) können ebenfalls genutzt werden, um den zu prüfenden Ausdruck zu erweitern. Mit dem AND-Operator kontrollieren Sie, ob beide Ausdrücke `true` sind, bevor der Code ausgeführt wird. Ein logisches OR dient dazu, sicherzustellen, dass einer oder mehrere der Ausdrücke `true` sind. Zudem können die logischen Operatoren durch den Einsatz des !-Operators (Invert) auf einen booleschen Wert umgekehrt werden.

Der typische Anwendungsfall für eine `if`-Anweisung ist das Treffen einer Entscheidung zwischen zwei oder mehr Optionen. Haben Sie nur zwei Optionen, ist diese Art von Ablaufsteuerung ideal.

Es gibt auch noch ein »Collection-if«, das zusätzliche Funktionalität zum Testen eines Elements mitbringt. Die mit dem Element verbundenen Daten lassen sich abhängig von Bedingungen im Collection-Objekt setzen – zum Beispiel, wenn es das erste oder letzte Objekt in einer Datenstruktur ist.

Iterieren, bis eine Bedingung erfüllt ist

Problem

Eine Methode soll laufen, bis eine Bedingung in einer Anwendung erfüllt ist.

Lösung

Nutzen Sie eine `while`-Schleife, wenn die Eintrittsbedingung zu Beginn des Programmablaufs überprüft werden soll. Die Prüfung findet also am Anfang der Schleife statt. Eine `while`-Schleife läuft also minimal gar nicht und maximal *N*-mal durch.

Dies ist ein Beispiel für eine Ablaufsteuerung durch eine `while`-Schleife:

```
void main() {

  bool isTrue = true;

  while (isTrue) {
    print ('Hallo');
    isTrue = false;
  }
}
```

Nutzen Sie eine `do while`-Schleife, wenn sie mindestens einmal ausgeführt werden soll.

Wie Sie in Abbildung 2-2 sehen, wird die Schleifenbedingung beim Eintritt in die Schleife überprüft – also vor jeder Iteration.

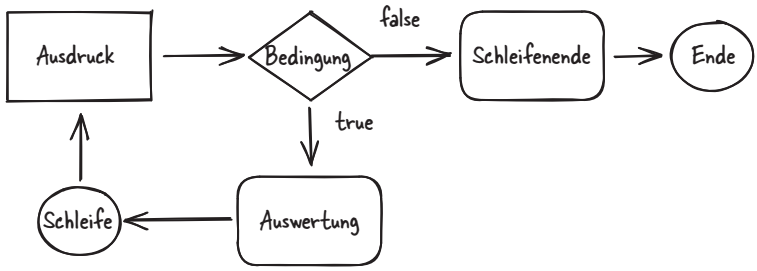

Abbildung 2-2: Logik einer `while`*-Schleife*

Hier ein Beispiel für eine Ablaufsteuerung mit einer `do while`-Schleife:

```
void main() {

  bool isTrue = true;

  do {
    print ('Hallo');
    isTrue = false;
  } while (isTrue) ;
}
```

Beachten Sie den Unterschied in Abbildung 2-3 zum vorigen Beispiel mit der `while`-Bedingung.

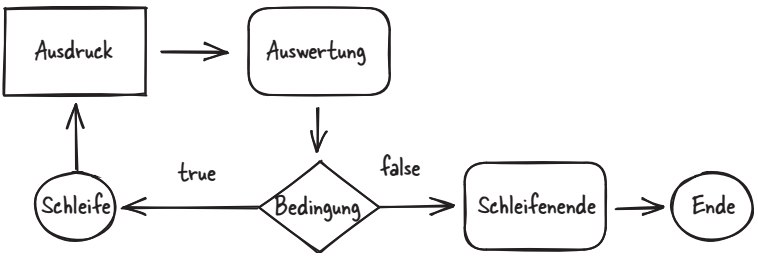

Abbildung 2-3: Logik einer `do while`*-Schleife*

Bei dieser Kontrollstruktur wird die Bedingung am Ende der Anweisungen ausgeführt – die Schleife läuft also mindestens einmal durch.

Diskussion

Der entscheidende Unterschied in diesen Beispielen ist die Art der Ausführung und was das für den Programmablauf bedeutet.

Im Beispiel mit der `while`-Schleife gibt die Demo-Anwendung nur einen Wert aus, wenn die `bool`-Variable auf `true` gesetzt ist. Das Beispiel mit der `do while`-Schleife führt auf jeden Fall eine `print`-Anweisung aus – unabhängig vom initialen Wert der Variablen `isTrue`.

Eine `while`-Schleife prüft eine Bedingung, bevor die Schleife ausgeführt wird; Sie können damit also 0 ... *N* Iterationen ablaufen lassen. Ein typischer Anwendungsfall besteht darin, eine Variable zu verwenden, um die Anzahl der Iterationen zu steuern.

Bei einer `do while`-Anweisung besteht der typische Anwendungsfall hingegen darin, die Schleife mindestens einmal laufen zu lassen. In solchen Situationen ist die Wahl einer `do while`-Schleife also besser.