

UNIVERSITÄT HEIDELBERG
SOMMERSEMESTER 2015

Softwareentwicklung für iOS

APP KATALOG

NILS FISCHER

Aktualisiert am 13. April 2015
Kursdetails und begleitende Materialien auf der Vorlesungswebseite:
<http://ios-dev-kurs.github.io>

Inhaltsverzeichnis

1	Über dieses Dokument	3
2	Hello World	4
2.1	Grundlagen der Programmierung in Swift	4

Kapitel 1

Über dieses Dokument

Dieser App Katalog enthält Schritt-für-Schritt Anleitungen für die im Rahmen unseres Kurses erstellten Apps sowie die wöchentlich zu bearbeitenden Übungsaufgaben und wird im Verlauf des Semesters kapitelweise auf der Vorlesungswebseite ^[1] zur Verfügung gestellt.

Er dient jedoch nur als Ergänzung zum parallel verfügbaren **Skript**, auf das hier häufig verwiesen wird. Dort sind die Erläuterungen zu den verwendeten Technologien, Methoden und Begriffen zu finden.

¹<http://ios-dev-kurs.github.io/>

Kapitel 2





Hello World

Was ist schon ein Programmierkurs, der nicht mit einem klassischen *Hello World* Programm beginnt? Wir werden jedoch noch einen Schritt weitergehen und diesen Gruß vom iOS Simulator oder, soweit vorhanden, direkt von unseren eigenen iOS Geräten ausgeben lassen. Außerdem wird in die objektorientierte Programmierung in *Swift* eingeführt.

Relevante Kapitel im Skript: Xcode, Programmieren in Swift sowie das Buch The Swift Programming Language [1]

2.1 Grundlagen der Programmierung in Swift

Anhand des ersten Kapitels *A Swift Tour* des Buches *The Swift Programming Language* lernen wir zunächst die Grundlagen der Programmierung in Swift kennen.

1. Öffnet Xcode und erstellt zunächst einen *Playground* mit  +  +  + . Playgrounds sind interaktive Skripte, mit denen sich ideal Code ausprobieren lässt. Gebt der Datei einen Namen wie "**01 – Grundlagen der Programmierung in Swift**" und speichert sie in einem Verzeichnis für diesen Kurs.
2. Ein Playground besteht aus einem Editor- und einem Inspektorbereich und führt geschriebenen Code automatisch aus. Ausgaben und Laufzeitinformationen werden im Inspektor angezeigt. In nur einer Zeile Code können wir den traditionellen *Hello World!*-Gruß ausgeben lassen (s. S. 5, Abb. 2.1).

```
1 println("Hello World!")
```

3. Nun lernen wir anhand des ersten Kapitels *A Swift Tour* des Buches *The Swift Programming Language* zunächst die Grundlagen der Programmierung in Swift. Macht euch dabei mit folgenden Begriffen vertraut:

¹https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/

- Variablen (**var**) und Konstanten(**let**)
- Einfache Datentypen (**Int**, **Float**, **Double**, **Bool**, **String**, **Array**, **Dictionary** und **Set**)
- Type Inference
- String-Formatierung
- Einfache Operatoren (+, -, *, /, \%)
- Abfragen (**if**, **switch**) und Schleifen (**for**, **while**)
- Optionals
- Funktionen

Im zweiten Kapitel *Language Guide* werden diese Konzepte noch einmal detailliert erklärt. Informiert euch dort gerne genauer darüber. Zunächst genügt es jedoch, einen Überblick zu erhalten. Im Verlauf des Kurses werden wir noch viel Übung im Umgang mit diesen Konzepten bekommen.

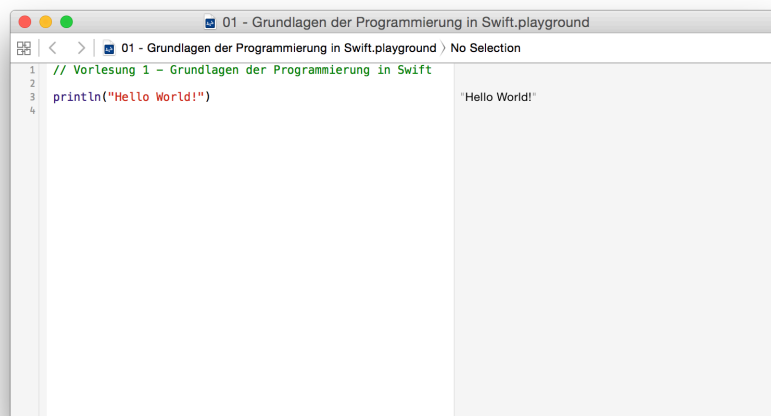


Abbildung 2.1: Playgrounds eignen sich ideal zum Ausprobieren von Swift Code.

Übungsaufgaben

1. Fibonacci

[1 P.]

- a) Schreibt einen Algorithmus, der alle Folgenglieder $F_n < 1000$ der Fibonaccifolge

$$F_n = F_{n-1} + F_{n-2} \quad (2.1)$$

$$F_1 = 1, F_2 = 2 \quad (2.2)$$

in der Konsole ausgibt.

- b) **Extra:** Bei jeder geraden Fibonaccizahl F_j ist der Abstand $\Delta n = j - i$ zum vorherigen geraden Folgenglied F_i auszugeben.

2. Primzahlen

[2 P.]

- a) Schreibt eine Funktion `primeNumbersUpTo:`, die ein Argument `maxNumber: Int` annimmt und alle Primzahlen bis `maxNumber` als Liste `[Int]` zurückgibt.

Hinweis: Mit dem Modulo-Operator `%` kann der Rest der Division zweier Integer gefunden werden:

```
1 let a = 20%3 // a ist jetzt Int(2)
```

- b) *Optionals* sind eines der elegantesten Konzepte in Swift, und sind auch in anderen modernen Sprachen zu finden. Informiert euch darüber im Kapitel *Language Guide > The Basics > Optionals* in *The Swift Programming Language*. Dieses Kapitel (bis einschließlich *Implicitly Unwrapped Optionals*) ist sehr wichtig, da wir in der iOS App Programmierung häufig mit Optionals arbeiten werden!
- c) Verwendet eure Liste von Primzahlen aus der vorigen Aufgabe, um effizienter zu prüfen, ob eine Zahl eine Primzahl ist.

Schreibt dazu eine Funktion `isPrimeNumber:cachedPrimeNumbers:`, die eine Zahl `n: Int` und eine **optionale** Liste von Primzahlen `cachedPrimeNumbers: [Int]?` annimmt. Verwendet die *Optional Binding Syntax* `if let` um mit dieser Liste zu arbeiten, wenn eine solche übergeben wurde und lang genug ist. Dann genügt es zu prüfen, ob die Zahl in der Liste enthalten ist. Wenn keine Liste übergeben wurde, soll die Primzahl wie in a) manuell geprüft werden.

Hinweise:

- Dem Argument `cachedPrimeNumbers` können wir einen *default* Wert `nil` zuweisen:

```
1 func isPrimeNumber(n: Int, cachedPrimeNumbers: [Int]? = nil) ->
  ↳ Bool {
2     // ...
3 }
```

So kann die Funktion auch ohne dieses Argument aufgerufen werden:

```
1 isPrimeNumber(7, cachedPrimeNumbers: [ 1, 2, 3, 7 ]) //
  ↳ vollständiger Funktionsaufruf, verwendet übergebene Liste zum
  ↳ Nachschlagen
2 isPrimeNumber(7) // äquivalent zu:
3 isPrimeNumber(7, cachedPrimeNumbers: nil) // Prüft Primzahl
  ↳ manuell
```

- Die globale Funktion `contains` prüft ob eine Element in einer Liste enthalten ist:

```
1 contains([ 1, 2, 3, 7 ], 7) // true
```

- Testet eure Funktion, indem Ihr bspw. folgenden Code ans Ende des Storyboards setzt:

```
1 //: ## Testing
2
3 let n = 499 // Number to test
4 let cachedMax = 500 // Prime numbers up to this number will be
   ↳ cached
5 import Foundation
6 var startDate: NSDate
7
8 startDate = NSDate()
9 let cachedPrimeNumbers = primeNumbersUpTo(cachedMax)
10 println("Time for caching prime numbers up to \(cachedMax):
   ↳ \(-startDate.timeIntervalSinceNow)s")
11
12 startDate = NSDate()
13 if isPrimeNumber(n) {
14     println("\(n) is a prime number.")
15 } else {
16     println("\(n) is not a prime number.")
17 }
18 let withoutCacheTime = -startDate.timeIntervalSinceNow
19 println("Time without cache: \(withoutCacheTime)s")
20
21 startDate = NSDate()
22 isPrimeNumber(n, cachedPrimeNumbers: cachedPrimeNumbers)
23 let withCacheTime = -startDate.timeIntervalSinceNow
24 println("Time with cache: \(withCacheTime)s (\((1 - withCacheTime
   ↳ / withoutCacheTime) * 100)% faster)")
```
