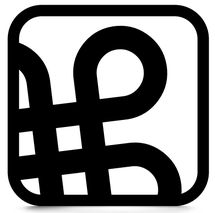


Macoun

Swift unter der Lupe

Natalia Ossipova
@nossipova

Warum programmieren wir nicht in Assembler?

Was ist Swift?

- Statisch typisiert
- Objektorientiert
- Imperativ
- Blockstruktur
- Funktional?

Was ist Swift?

Beeinflusst von

Objective-C	Haskell	Rust
Ruby	Python	C#
CLU	…	

Spachkonzepte

Typableitung (Type Inference)

Herleitung des Typs eines Wertes durch den Compiler

Typableitung: Literale

```
"Macoun"            // String

42                  // Int

3.14159             // Double

true                // Bool
```

Typableitung: Ausdrücke

```
5 + 2             // Int

23.68 + 54.97     // Double
```

Typableitung: Variablen

```
let a = 10          // Int

var b = 20          // Int

b = a + b           // Int
```

Typerzwingung (Type Coercion)

```
15 + 27.14          // Double
```

Typableitung: Integer

Integer-Literal hat keinen expliziten Typ

```
// Demo: Type Inference 1
```

Typableitung: Array

Array-Typ beim Initialisieren

```
// Demo: Type Inference 2
```

Typumwandlung (Type Casting)

- Immer explizit
- Nur für Subklassen, Protokolle und Any/AnyObject

```
34.7 as Int     // Compilezeitfehler
```

Typumwandlung für Arrays

```
let daysOfWeek: Any = [“Mon”, “Tue”, “Wed”,
                       “Thur”, “Fri”, “Sat”, “Sun”]

daysOfWeek as [String]
```

Tupel

- Eindimensionales Array
- Unterschiedliche Typen
- Rückgabe mehrerer Werte aus Funktionen

Dekomposition von Tupeln

```
let (length, unit) = (15, "km")

let distance = (15, "km")
let number = distance.0
```

Dekomposition von Tupeln

```
let macoun = (latitude: 50.106996, longitude: 8.689346)
let latitudeMacoun = macoun.latitude

let (_, longitudeMacoun) = macoun
```

Ausdruck vs. Anweisung

Ausdruck

```
let value = someVariable < 10 ? 14 : 7
```

Ausdruck vs. Anweisung

Anweisung

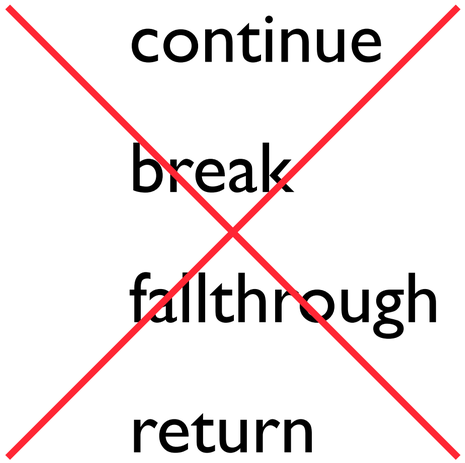
```
var value: Int

switch someVariable {
case 10: value = 14
case 15: value = 28
default: value = 7
}
```

Ausdrücke statt Anweisungen

Was wäre wenn…?

```
let value = switch someVariable {    // Zur Zeit nicht möglich!
case 10: 14
case 15: 28
default: 7
}
```

Sprunganweisungen



Remember goto fail;

Switch-Anweisung

- Musterabgleich (Pattern Matching)

  Werte Wertebereiche Tupel

  Typ where-Bedingungen

- Keine implizite Übertragung der Steuerung (fallthrough)
- Reihenfolge ist wichtig!

Operatorüberladung

- Operator ist eine globale Funktion
- Kontrovers: Verkettung von Zeichenfolgen

```
42 + 3.14159 == 3.14159 + 42      // Kommutativ

42 + “Macoun” != “Macoun” + 42    // Nicht kommutativ
```

Operatorüberladung

Mittel für interne domänenspezifische Sprachen

```
// Demo: Operator Overloading
```

Funktionen

- First-Class-Objekte
- Funktionstyp: Typ der Parameter + Typ des Rückgabewertes

```
// Demo: Functions
```

Closure

- anonyme Funktion, Lambda
- Ausführung außerhalb des ursprünglichen Scope
- Aufrufkontext

Closure

- Variablenbindung per Referenz

```
// Demo: Closures
```

Generische Typen

- Funktionen
- Typen: Klassen, Strukturen, Enumerations

Generische Typen

- Typ-Constraints

```
// Demo: Type Constraints
```

Ist Swift funktional?

Funktionale Programmierung

- Auswertung von Ausdrücken, keine Anweisungen
- Zustandslos, keine veränderbaren Daten
- Referenzielle Transparenz: gleiche Eingabewerte, gleiches Ergebnis
- Keine Nebeneffekte
- Deklarativ: “was” statt “wie”

Nicht-funktional

- Veränderbarkeit
  - Variablen
  - Eigenschaften von Objekten in einem Array immer veränderbar
  - Variablenbindung per Referenz in Closures
  - inout-Parameter
- Anweisungen

Funktional programmieren

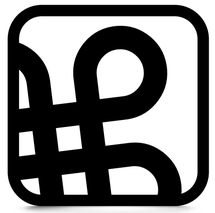
```
// Demo: Fibonacci-Folge: 1, 1, 2, 3, 5, 8, 13, 21, …
```

Swift

- Funktionale Programmiersprache? ✗
- Funktional programmieren? ✓

Fragen?

Vielen Dank



Macoun