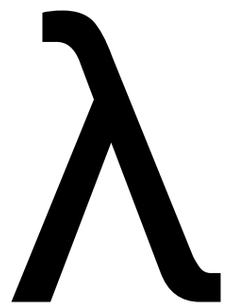

# Funktionale Programmierung mit Swift

# Swift Paradigmen

- Objektorientierte Programmierung
- Imperative Programmierung
- Funktionale Programmierung

"…functional programming is a programming paradigm… that treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data."

–Wikipedia

# Mathematische Funktion

Jede Ausführung mit den gleichen Argumenten erzeugt immer das gleiche Ergebnis.

# Merkmale funktionaler Programmierung

- keine Seiteneffekte
- Funktionen höherer Ordnung
- Nichtveränderbarkeit von Daten
- sehr gut testbar

# Keine Seiteneffekte

- Funktionen liefern nur Ergebnisse auf Basis der Eingabeparameter
- Ausführung erzeugt keine Änderung am System

# Funktionen höherer Ordnung

- Funktionen, die **Funktionen** als Argumente **erhalten** oder
- Funktionen, die **Funktionen** als Ergebnis **liefern**

# Demo

# Funktionen höherer Ordnung in der Standard Library

- **reduce**
  Kombiniert alle Werte einer Collection zu einem Wert

- **map**
  erzeugt eine neue Collection mit einem neuen Wert für jeden Wert in der Eingabe-Collection

- **filter**
  Gibt nur die Werte einer Collection zurück, die einem gegebenen Kriterium entsprechen

# Value Types

Initialisierung, Zuweisungen und Weitergabe an Funktionen erzeugen unabhängige Kopien der Werte.

"… almost all the types in Swift are value types, including arrays, dictionaries, numbers, booleans, tuples, and enums… Classes are the exception, rather than the rule"

Excerpt from: "Functional Programming in Swift."

# Beispiel

http://nomothetis.svbtle.com/error-handling-in-swift

```
NSError *error = nil;
CGFloat result = [GHCalculator divide:2.5 by:0 error:&error];

if (error) {
    NSLog(@"An error occurred: %@", error);
} else {
    NSLog(@"The result is %g", result);
}
```

```
enum Result {
    case Value(AnyObject)
    case Error(String)
}
```

```
let result = GHCalculator.divide(2.5, by: 0)

switch result {
case .Value(let quotient):
    println("The result is \(quotient).")
case .Error(let errorMessage):
    println(errorMessage)
}
```

# Demo

## Functional Filter Chaining

# Fragen?

# Quellen

- Vortrag von Natasha Murashev (@NatashaTheRobot) auf der Functional Swift Conference Anfang Dezember 2014

- “Functional Programming in Swift” (Chris Eidhof, et al.)

- “Error Handling in Swift: Might and Magic” von Alexandros Salazar (http://nomothetis.svbtle.com/error-handling-in-swift)

- weitere Vorträge der Functional Swift Conference