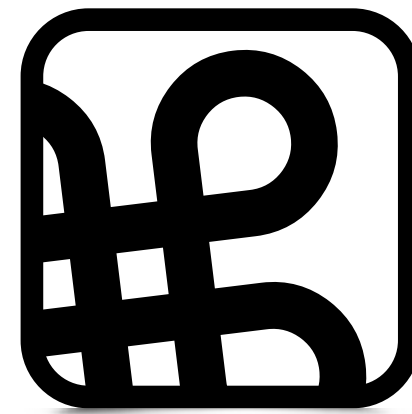


ortierter Autor 21.07.2015, 13:13  
bereitung:

Demos, Files im Finder anordnen.

Alle Programme die man für Demos benötigt vorher(!) starten.  
sw: Xcode, Terminal, Refelection.

Präsentation starten und auf Zeichen oder Signal warten.



# Macoun

# Dynamisches Swift

Nikolaj Schumacher

# Was ist eine dynamische Sprache

nicht statisch

# Dynamic programming language

From Wikipedia, the free encyclopedia

*This article is about a class of programming languages. For the method for reducing the running time of algorithms, see Dynamic programming (algorithm).*

**This article has multiple issues.** Please help [improve it](#) or discuss these issues on the [talk page](#) ([view instructions](#) and [learn how to edit](#) [Wikipedia](#) [messages](#))



- This article **needs attention from an expert on the subject**. (*January 2015*)
- This article's **factual accuracy is [disputed](#)**. (*March 2012*)
- This article **may be [confusing or unclear](#) to readers**. (*October 2009*)

# Was ist eine statische Sprache

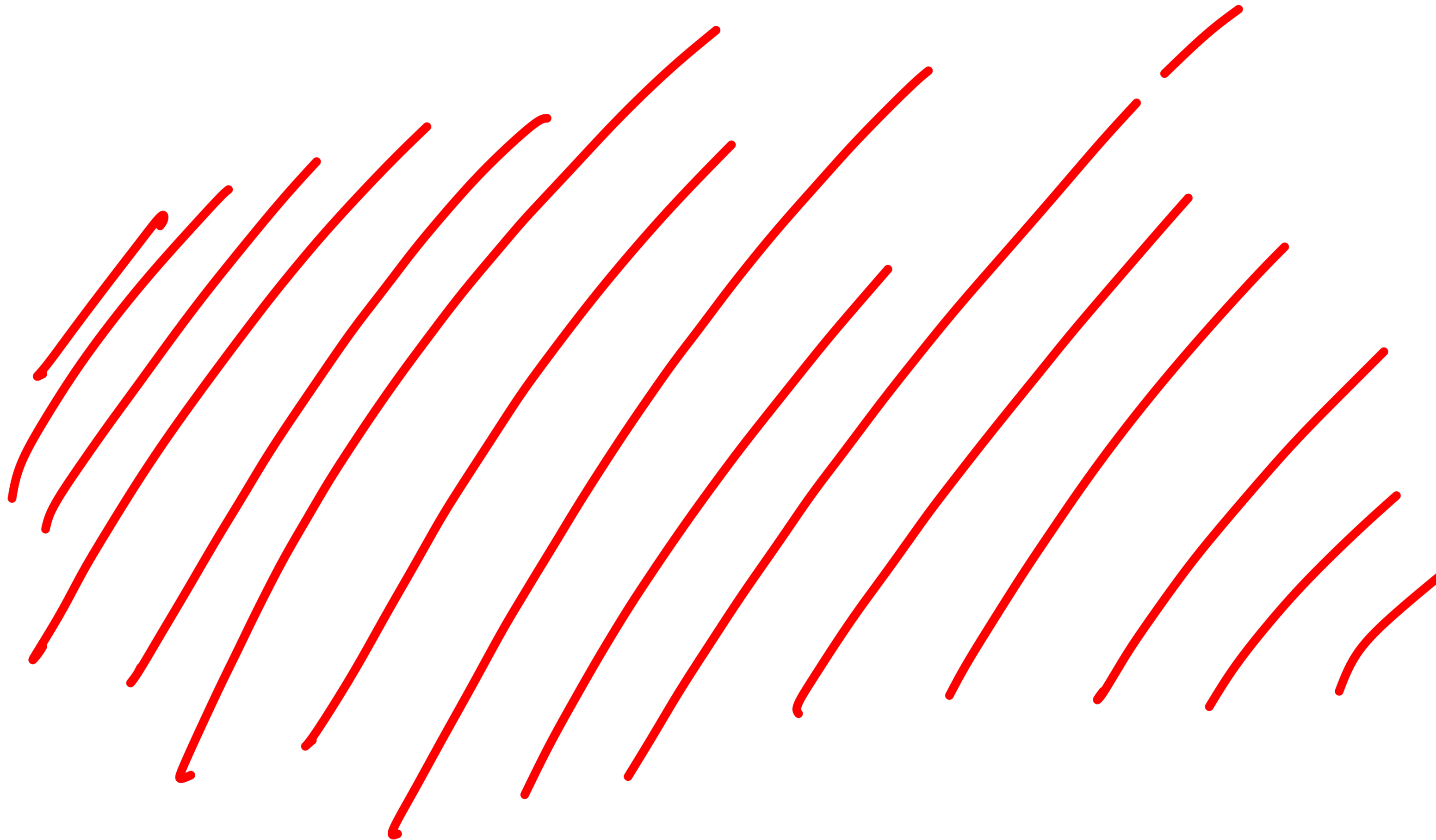
Der Compiler macht „mehr“

# Was ist eine statische Sprache

- Type Checking
- Statische Analysen
- ...
- entfernt Abstraktionen

statisch ↔ dynamisch

C



Lisp  
JavaScript

# dynamische Features

- Typisierung
  - stark oder schwach
  - Duck Typing
  - statisch oder dynamisch



# dynamische Features

- Method Dispatch
  - early oder late Binding

# dynamische Features

- Reflection

# dynamische Features

- Metaprogrammierung

# dynamische Features

- Evaluierung zur Laufzeit
  - "eval is evil"

# dynamische Features

- ~~langsam~~

# Wo steht ObjC?

Feature	ObjC
Typisierung	schwach stark
statische Typisierung	optional
Method Dispatch	statisch virtual dynamisch
Reflection	✓
eval	-
langsam	manchmal

# Wo steht Swift?

Feature	Swift (mit ObjC Runtime)
Typisierung	schwach stark
statische Typisierung	optional
Method Dispatch	statisch virtual dynamisch
Reflection	✓
eval	-
langsam	manchmal

# Swift ohne ObjC-Runtime

- kein NSObject
  - kein KVO
  - kein performSelector
  - kein valueForKeyPath
- kein dynamic



```
let x: Any = 5
```

```
func printType(_ x: Any) {  
    if x is Int {  
        print("Int")  
    }  
}
```

```
let x: Any = 5  
print("\(type(of: x))")
```

```
func describe(_ x: Any) {  
    if let x = x as? CustomStringConvertible {  
        print(x.description)  
    }  
}
```

```
protocol Animal {  
    func walk()  
  
    // optional:  
    func quack()  
}
```

```
extension Animal {  
    func quack() {}  
}
```

```
class Dog: Animal {  
    func walk() {  
        // Swift doesn't have goto  
    }  
}
```

```
class Duck: Animal {  
    func walk() {}  
  
    func quack() {  
        print("quack")  
    }  
}
```

```
protocol Animal {  
    func walk()  
    func quack()  
}
```

```
protocol Quacks: Animal {  
    func quack()  
}
```

```
class Dog: Animal {  
    func walk() {  
        // Swift doesn't have goto  
    }  
}
```

```
class Duck: Quacks {  
    func walk() {}  
  
    func quack() {  
        print("quack")  
    }  
}
```

```
class Animal {  
    func quack() {}  
}  
class Duck {  
    func quack() {  
        print("quack")  
    }  
}  
  
let animal: Animal  
animal.quack()
```

# Warum?



- Metadata
- Nominal Type Descriptor
- vtables, witness tables
- <https://github.com/apple/swift/blob/master/docs/ABI/TypeMetadata.rst>



# Mirror/Reflection

- ersetzt
  - KVC
  - ObjC Reflection APIs (`class_copyPropertyList` & co)

Demo

# Codable

- nur Serialisierung?
- ersetzt auch KVC + ObjC Reflection APIs
- Schreibzugriff

Demo

# Dynamic Member Lookup

- Swift 4.2
- Ziel: Kompatibilität zu dynamischen Sprachen

# Dynamic Member Lookup

```
animal.legCount = a
```

```
animal[dynamicMember: "legCount"] = a
```

Demo

# Dynamic Callable

- Swift 5
- SE-0216
- Erweiterung von Dynamic Member Lookup für Funktionsaufrufe
- (fast) dynamic Dispatch



# Dynamic Callable

```
a = someValue(keyword1: 42, "foo", keyword2: 19)
```

```
a = someValue.dynamicallyCall(withKeywordArguments: [  
    "keyword1": 42, "" : "foo", "keyword2": 19  
])
```

# Was funktioniert nicht

- Method Swizzling

# Was fehlt?

- dynamische Member können nicht
  - Protokolle implementieren
  - echte Methoden überschreiben

} Mocking

# Was fehlt?

- Methoden können nicht
  - per Namen gefunden werden
  - indirekt aufgerufen werden (a la NSInvocation)

# Was fehlt?

- Typen können nicht
  - per Namen gefunden werden (z.B. Interface Builder)
  - ohne Instanz reflektiert werden

# Was fehlt?

- Mirrors können nicht
  - Key-Paths liefern

# Was fehlt?

- APIs
  - Informationen sind da
  - externe Tools möglich

# Wo steht Swift?

Feature	Swift
Typisierung	schwach stark
statische Typisierung	Opt-out
Method Dispatch	statisch virtual dynamisch
Reflection	✓
eval	-
langsam	manchmal



# Schlussfolgerung



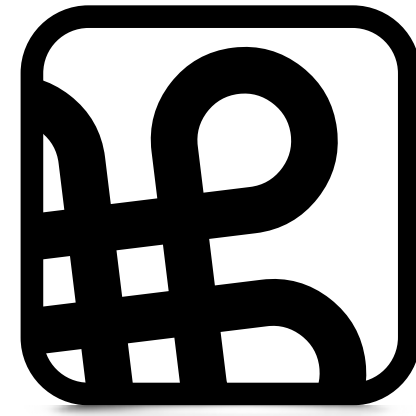
# Fragen?

<https://github.com/nschum/talks>



Vielen Dank

<https://github.com/nschum/talks>



**Macoun**