

# INTRODUCTION TO SWIFT

---

*Lisa Obermaier, Simon Thum*  
*03 May 2019*

*Mobile Application Development*  
*Prof. Dr. Gudrun Socher*

- Motivations of Swift
- Technical Background
- iOS App Development with Swift
- Live Programming
- Summary

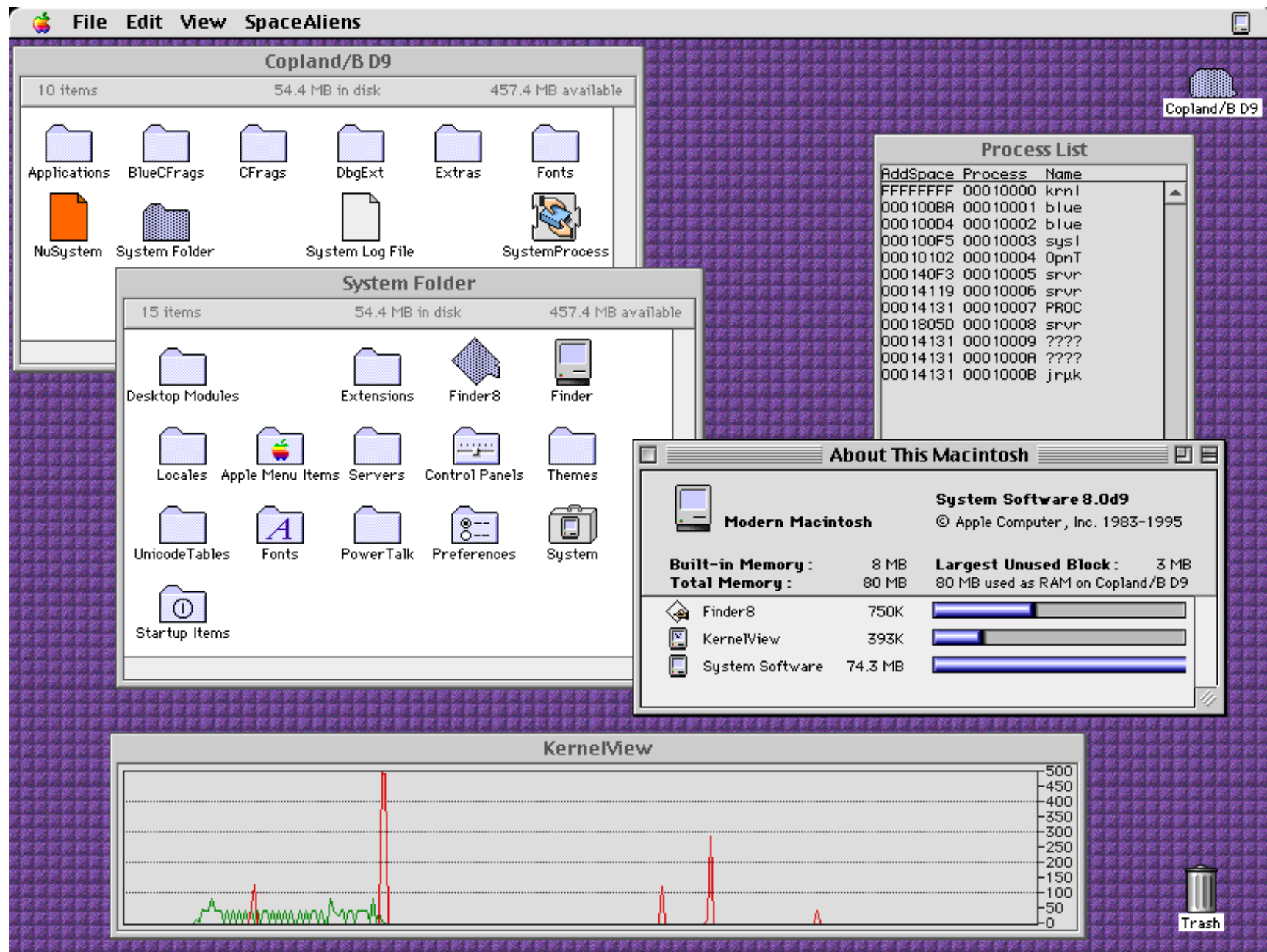
# MOTIVATIONS FOR SWIFT

---

*(for Apple)*

**"AVOIDING  
COPLAND 2010"**

# PROJECT COPLAND



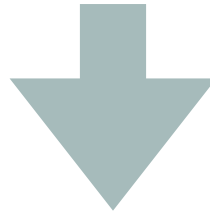
# **MEMORY-MANAGED LANGUAGE / API**

# ENTER SWIFT

---

*Obj-C*

```
NSString *str = @"Hello,";  
str = [str stringByAppendingString:@" world."];
```



*Swift*

```
var str = "Hello,"  
str += " world."
```

# TECHNICAL BACKGROUND










---

*(for Developers)*



# SWIFT AT A GLANCE

---

- Imperative  like C, not like HTML
- ...and Functional, too  "you can use recursion"
- Block-structured  {}
- Object-oriented
- Protocol-oriented  like Interfaces
- Type Inference  `var x = 1`
- Static Typing  `x = "abc"`  *error: x has been inferred as an int*
- Dot-Notation  `let centerX = origin.x + (size.width / 2)`
- UTF-8  `let cat = "🐱"`

# BEYOND THE BASICS

---

- Error Handling
- Assertions
- Generics
- Extensions
- Closures
- Optionals
- Property Observers
- Automated Reference Counting

# ERROR HANDLING

---

```
do {  
    try makeASandwich()  
    eatASandwich()  
} catch SandwichError.outOfCleanDishes {  
    washDishes()  
} catch SandwichError.missingIngredients(let ingredients) {  
    buyGroceries(ingredients)  
}
```

# ASSERTIONS

---

```
let age = -3
assert(age >= 0, "A person's age can't be less than zero.")
// This assertion fails because -3 is not >= 0.
```

# GENERIC

---

```
func swapTwoStrings(_ a: inout String, _ b: inout String) {  
    let temporaryA = a  
    a = b  
    b = temporaryA  
}  
  
func swapTwoDoubles(_ a: inout Double, _ b: inout Double) {  
    let temporaryA = a  
    a = b  
    b = temporaryA  
}
```

```
func swapTwoValues<T>(_ a: inout T, _ b: inout T) {  
    let temporaryA = a  
    a = b  
    b = temporaryA  
}
```

# EXTENSIONS

---

```
protocol StringConvertible {  
    func toString() -> String  
}  
  
extension String: StringConvertible {  
    func toString() -> String {  
        return self  
    }  
}  
  
var thisMustHaveAToString: StringConvertible  
  
/* ... */  
  
print(thisMustHaveAToString.toString())
```

# CLOSURES

---

```
let names = ["Chris", "Alex", "Ewa", "Barry", "Daniella"]

func backward(_ s1: String, _ s2: String) -> Bool {
    return s1 > s2
}

var reversedNames = names.sorted(by: backward)
// reversedNames is equal to ["Ewa", "Daniella", "Chris", "Barry", "Alex"]
```

```
reversedNames = names.sorted(by: { (s1: String, s2: String) -> Bool in
    return s1 > s2
})
```

```
reversedNames = names.sorted(by: { s1, s2 in s1 > s2 } )
```

```
reversedNames = names.sorted(by: >)
```

# OPTIONALS

---

```
let possibleNumber = "123"  
let convertedNumber = Int(possibleNumber)  
// convertedNumber is inferred to be of type "Int?", or "optional Int"
```

```
if convertedNumber != nil {  
    print("convertedNumber has an integer value of \$(convertedNumber!).")  
}  
// Prints "convertedNumber has an integer value of 123."
```



# PROPERTY OBSERVERS

---

```
class StepCounter {  
    var totalSteps: Int = 0 {  
        willSet(newTotalSteps) {  
            print("About to set totalSteps to \$(newTotalSteps)")  
        }  
        didSet {  
            if totalSteps > oldValue {  
                print("Added \$(totalSteps - oldValue) steps")  
            }  
        }  
    }  
}  
  
let stepCounter = StepCounter()  
stepCounter.totalSteps = 200  
// About to set totalSteps to 200  
// Added 200 steps
```

# AUTOMATED REFERENCE COUNTING

---

```
class Person {  
  let name: String  
  init(name: String) {  
    self.name = name  
    print("\(name) is being initialized")  
  }  
  deinit {  
    print("\(name) is being deinitialized")  
  }  
}
```

```
var reference1: Person?  
var reference2: Person?  
var reference3: Person?
```

```
reference1 = Person(name: "John Appleseed")  
// Prints "John Appleseed is being initialized"
```

```
reference2 = reference1  
reference3 = reference1
```

```
reference1 = nil  
reference2 = nil
```

```
reference3 = nil  
// Prints "John Appleseed is being deinitialized"
```

# **IOS APP DEVELOPMENT**

# THAT USUALLY MEANS

---

- Writing Swift
  - in XCode
    - Calling CocoaTouch APIs
      - Using the UIKit Framework

**ACTUALLY, LET'S JUST  
DO IT TOGETHER**

# TO SUM IT ALL UP

---

*for Apple and the Rest of Us*