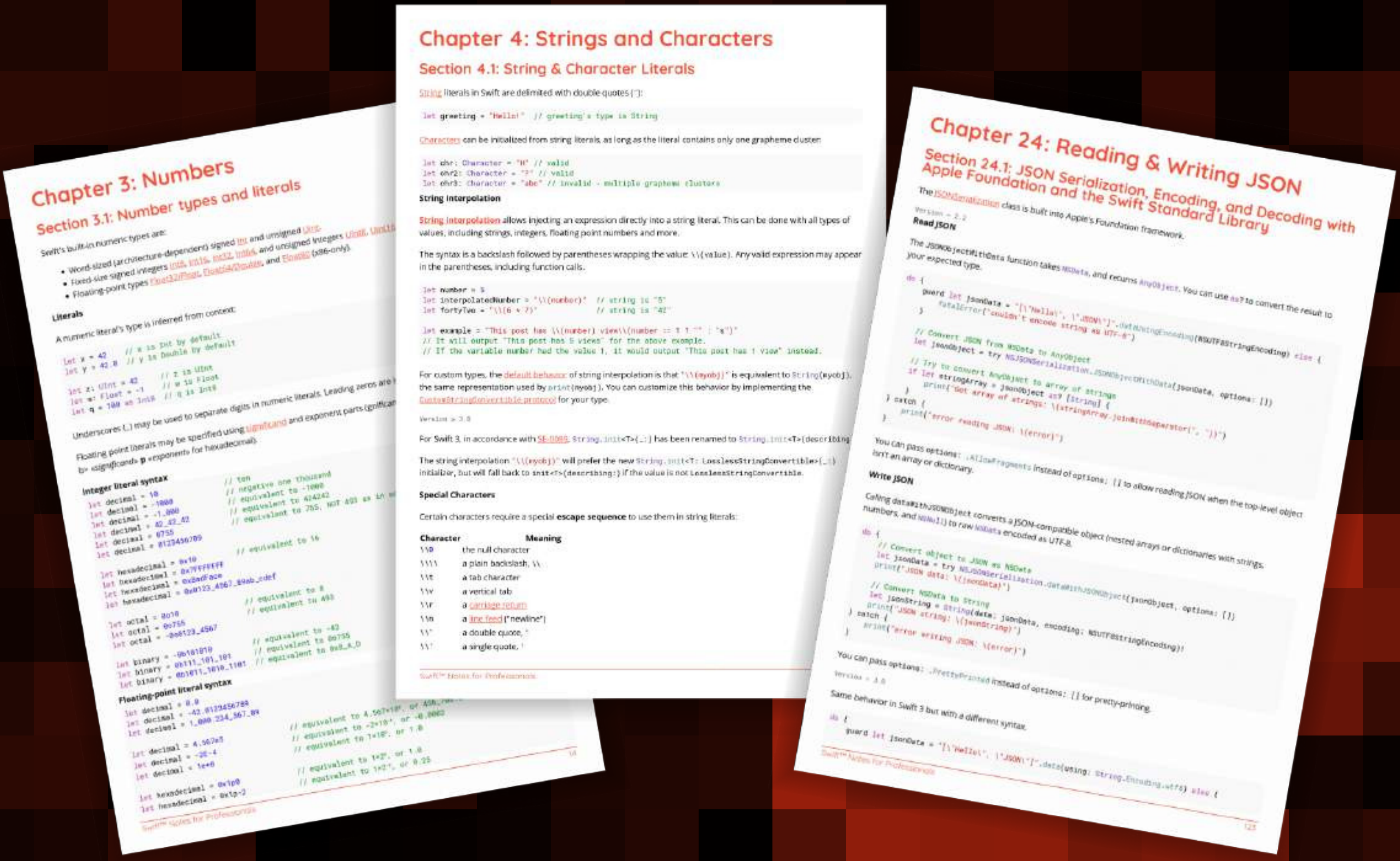
Swift™
Notes for Professionals



Chapter 3: Numbers

Section 3.1: Number types and literals

Chapter 4: Strings and Characters

Section 4.1: String & Character Literals

String literals in Swift are delimited with double quotes ("):

```
let greeting = "Hello!"  // greeting's type is String
```

Characters can be initialized from string literals, as long as the literal contains only one grapheme cluster:

```
let chr: Character = "H" // valid
let chr2: Character = "?" // valid
let chr3: Character = "abc" // invalid - multiple grapheme clusters
```

String interpolation

String interpolation allows injecting an expression directly into a string literal. This can be done with all types of values, including strings, integers, floating point numbers and more.

The syntax is a backslash followed by parentheses wrapping the value: \\(value). Any valid expression may appear in the parentheses, including function calls.

```
let number = 5
let interpolatedNumber = "\\(number)"  // string is "5"
let fortyTwo = "\\(6 * 7)"             // string is "42"

let example = "This post has \\(number) view\\(number == 1 ? "" : "s")"
// It will output "This post has 5 views" for the above example.
// If the variable number had the value 1, it would output "This post has 1 view" instead.
```

For custom types, the default behavior of string interpolation is that "\\(myobj)" is equivalent to String(myobj), the same representation used by print(myobj). You can customize this behavior by implementing the CustomStringConvertible protocol for your type.

Version ≥ 3.0

For Swift 3, in accordance with SE-0089, String.init<T>(_:) has been renamed to String.init<T>(describing

The string interpolation "\\(myobj)" will prefer the new String.init<T: LosslessStringConvertible>(_:) initializer, but will fall back to init<T>(describing:) if the value is not LosslessStringConvertible.

Special Characters

Certain characters require a special **escape sequence** to use them in string literals:

Character	Meaning
\\0	the null character
\\\\	a plain backslash, \\
\\t	a tab character
\\v	a vertical tab
\\r	a carriage return
\\n	a line feed ("newline")
\\"	a double quote, "
\\'	a single quote, '

Swift™ Notes for Professionals

Chapter 24: Reading & Writing JSON

Section 24.1: JSON Serialization, Encoding, and Decoding with Apple Foundation and the Swift Standard Library

200+ pages

of professional hints and tricks

GoalKicker.com
Free Programming Books

Disclaimer
This is an unofficial free book created for educational purposes and is not affiliated with official Swift™ group(s) or company(s).
All trademarks and registered trademarks are the property of their respective owners

Contents

About 1

Chapter 1: Getting started with Swift Language 2

- Section 1.1: Your first Swift program 2
- Section 1.2: Your first program in Swift on a Mac (using a Playground) 3
- Section 1.3: Your first program in Swift Playgrounds app on iPad 7
- Section 1.4: Installing Swift 8
- Section 1.5: Optional Value and Optional enum 8

Chapter 2: Variables & Properties 10

- Section 2.1: Creating a Variable 10
- Section 2.2: Property Observers 10
- Section 2.3: Lazy Stored Properties 11
- Section 2.4: Property Basics 11
- Section 2.5: Computed Properties 12
- Section 2.6: Local and Global Variables 12
- Section 2.7: Type Properties 13

Chapter 3: Numbers 14

- Section 3.1: Number types and literals 14
- Section 3.2: Convert numbers to/from strings 15
- Section 3.3: Rounding 15
- Section 3.4: Random number generation 16
- Section 3.5: Convert one numeric type to another 17
- Section 3.6: Exponentiation 17

Chapter 4: Strings and Characters 18

- Section 4.1: String & Character Literals 18
- Section 4.2: Concatenate strings 19
- Section 4.3: String Encoding and Decomposition 20
- Section 4.4: Examine and compare strings 20
- Section 4.5: Reversing Strings 21
- Section 4.6: Check if String contains Characters from a Defined Set 21
- Section 4.7: String Iteration 22
- Section 4.8: Splitting a String into an Array 24
- Section 4.9: Unicode 24
- Section 4.10: Converting Swift string to a number type 25
- Section 4.11: Convert String to and from Data / NSData 25
- Section 4.12: Formatting Strings 26
- Section 4.13: Uppercase and Lowercase Strings 26
- Section 4.14: Remove characters from a string not defined in Set 27
- Section 4.15: Count occurrences of a Character into a String 27
- Section 4.16: Remove leading and trailing WhiteSpace and NewLine 27

Chapter 5: Booleans 29

- Section 5.1: What is Bool? 29
- Section 5.2: Booleans and Inline Conditionals 29
- Section 5.3: Boolean Logical Operators 30
- Section 5.4: Negate a Bool with the prefix ! operator 30

Chapter 6: Arrays 31

- Section 6.1: Basics of Arrays 31
- Section 6.2: Extracting values of a given type from an Array with flatMap(_:) 32

Section 6.3: Combining an Array's elements with reduce(_:combine:) 32
Section 6.4: Flattening the result of an Array transformation with flatMap(_:) 33
Section 6.5: Lazily flattening a multidimensional Array with flatten() 33
Section 6.6: Filtering out nil from an Array transformation with flatMap(_:) 34
Section 6.7: Subscripting an Array with a Range 34
Section 6.8: Removing element from an array without knowing it's index 35
Section 6.9: Sorting an Array of Strings 35
Section 6.10: Accessing indices safely 36
Section 6.11: Filtering an Array 37
Section 6.12: Transforming the elements of an Array with map(_:) 37
Section 6.13: Useful Methods 38
Section 6.14: Sorting an Array 38
Section 6.15: Finding the minimum or maximum element of an Array 39
Section 6.16: Modifying values in an array 40
Section 6.17: Comparing 2 Arrays with zip 40
Section 6.18: Grouping Array values 41
Section 6.19: Value Semantics 42
Section 6.20: Accessing Array Values 42

Chapter 7: Tuples 44
Section 7.1: What are Tuples? 44
Section 7.2: Decomposing into individual variables 44
Section 7.3: Tuples as the Return Value of Functions 45
Section 7.4: Using a typealias to name your tuple type 45
Section 7.5: Swapping values 46
Section 7.6: Tuples as Case in Switch 46

Chapter 8: Enums 48
Section 8.1: Basic enumerations 48
Section 8.2: Enums with associated values 48
Section 8.3: Indirect payloads 49
Section 8.4: Raw and Hash values 50
Section 8.5: Initializers 51
Section 8.6: Enumerations share many features with classes and structures 52
Section 8.7: Nested Enumerations 53

Chapter 9: Structs 54
Section 9.1: Structs are value types 54
Section 9.2: Accessing members of struct 54
Section 9.3: Basics of Structs 54
Section 9.4: Mutating a Struct 55
Section 9.5: Structs cannot inherit 55

Chapter 10: Sets 57
Section 10.1: Declaring Sets 57
Section 10.2: Performing operations on sets 57
Section 10.3: CountedSet 58
Section 10.4: Modifying values in a set 58
Section 10.5: Checking whether a set contains a value 58
Section 10.6: Adding values of my own type to a Set 58

Chapter 11: Dictionaries 60
Section 11.1: Declaring Dictionaries 60
Section 11.2: Accessing Values 60
Section 11.3: Change Value of Dictionary using Key 61

Section 11.4: Get all keys in Dictionary 61
Section 11.5: Modifying Dictionaries 61
Section 11.6: Merge two dictionaries 62
Chapter 12: Switch 63
Section 12.1: Switch and Optionals 63
Section 12.2: Basic Use 63
Section 12.3: Matching a Range 63
Section 12.4: Partial matching 64
Section 12.5: Using the where statement in a switch 65
Section 12.6: Matching Multiple Values 65
Section 12.7: Switch and Enums 66
Section 12.8: Switches and tuples 66
Section 12.9: Satisfy one of multiple constraints using switch 67
Section 12.10: Matching based on class - great for prepareForSegue 67
Section 12.11: Switch fallthroughs 68
Chapter 13: Optionals 69
Section 13.1: Types of Optionals 69
Section 13.2: Unwrapping an Optional 69
Section 13.3: Nil Coalescing Operator 71
Section 13.4: Optional Chaining 71
Section 13.5: Overview - Why Optionals? 72
Chapter 14: Conditionals 74
Section 14.1: Optional binding and "where" clauses 74
Section 14.2: Using Guard 75
Section 14.3: Basic conditionals: if-statements 75
Section 14.4: Ternary operator 76
Section 14.5: Nil-Coalescing Operator 77
Chapter 15: Error Handling 78
Section 15.1: Error handling basics 78
Section 15.2: Catching different error types 79
Section 15.3: Catch and Switch Pattern for Explicit Error Handling 80
Section 15.4: Disabling Error Propagation 81
Section 15.5: Create custom Error with localized description 81
Chapter 16: Loops 83
Section 16.1: For-in loop 83
Section 16.2: Repeat-while loop 85
Section 16.3: For-in loop with filtering 85
Section 16.4: Sequence Type forEach block 86
Section 16.5: while loop 86
Section 16.6: Breaking a loop 87
Chapter 17: Protocols 88
Section 17.1: Protocol Basics 88
Section 17.2: Delegate pattern 90
Section 17.3: Associated type requirements 91
Section 17.4: Class-Only Protocols 93
Section 17.5: Protocol extension for a specific conforming class 94
Section 17.6: Using the RawRepresentable protocol (Extensible Enum) 94
Section 17.7: Implementing Hashable protocol 95
Chapter 18: Functions 97

Section 18.1: Basic Use ... 97
Section 18.2: Functions with Parameters ... 97
Section 18.3: Subscripts ... 98
Section 18.4: Methods ... 99
Section 18.5: Variadic Parameters ... 100
Section 18.6: Operators are Functions ... 100
Section 18.7: Passing and returning functions ... 101
Section 18.8: Function types ... 101
Section 18.9: Inout Parameters ... 101
Section 18.10: Throwing Errors ... 101
Section 18.11: Returning Values ... 102
Section 18.12: Trailing Closure Syntax ... 102
Section 18.13: Functions With Closures ... 103

Chapter 19: Extensions ... 105

Section 19.1: What are Extensions? ... 105
Section 19.2: Variables and functions ... 105
Section 19.3: Initializers in Extensions ... 106
Section 19.4: Subscripts ... 106
Section 19.5: Protocol extensions ... 106
Section 19.6: Restrictions ... 107
Section 19.7: What are extensions and when to use them ... 107

Chapter 20: Classes ... 109

Section 20.1: Defining a Class ... 109
Section 20.2: Properties and Methods ... 109
Section 20.3: Reference Semantics ... 109
Section 20.4: Classes and Multiple Inheritance ... 110
Section 20.5: deinit ... 111

Chapter 21: Type Casting ... 112

Section 21.1: Downcasting ... 112
Section 21.2: Type casting in Swift Language ... 112
Section 21.3: Upcasting ... 114
Section 21.4: Example of using a downcast on a function parameter involving subclassing ... 114
Section 21.5: Casting with switch ... 115

Chapter 22: Generics ... 116

Section 22.1: The Basics of Generics ... 116
Section 22.2: Constraining Generic Placeholder Types ... 117
Section 22.3: Generic Class Examples ... 118
Section 22.4: Using Generics to Simplify Array Functions ... 119
Section 22.5: Advanced Type Constraints ... 119
Section 22.6: Generic Class Inheritance ... 120
Section 22.7: Use generics to enhance type-safety ... 121

Chapter 23: OptionSet ... 122

Section 23.1: OptionSet Protocol ... 122

Chapter 24: Reading & Writing JSON ... 123

Section 24.1: JSON Serialization, Encoding, and Decoding with Apple Foundation and the Swift Standard Library ... 123
Section 24.2: SwiftyJSON ... 126
Section 24.3: Freddy ... 127
Section 24.4: JSON Parsing Swift 3 ... 129
Section 24.5: Simple JSON parsing into custom objects ... 131

Click here to download full PDF material