Swift 4 - **Core Language**

This may be the weirdest part of Your journey. Unfortunately, it is also the most basic - and forms the basis for everything else. Understanding the programming language You use is of enormous importance. So, please try to understand the basic structure. Note that everything of meaning in Swift is Tokens. Tokens form Statements - and Statements exist in different forms. Simple Statements are whatever You typically use. Note Simple Statements!

Comments

Single-Line Comments

Single-Line Comments end at a newline character. They can also appear at the end of a line of code to comment on it:

```
// Comment
for sample in samples {   // Iterating through a Collections
```

Multiple-Line Comments

Multiple-Line Comments start with an opening character sequence: „/*“ and end at the closing character sequence: „*/“. Newline characters inside those character sequences are ignored. They can appear at the end of a line of code:

```
for sample in samples {   /* Iterating through a Collections
of items called „samples“ */
```

Tokenizables

Tokens are the most basic form of a meaningful piece of code: the minimum meaningful element in Your code. They are important, because they are the first thing that the compiler determines during its run - and as such, understanding how the compiler tokenizes Your code immensely helps understand the result. This part, though so very significant, is only rarely discussed (particularly in entry level texts - but even in advanced texts). I find them fascinating. Please spend some time thinking about this.
The tokenizer is the first component of the Swift compiiler that interprets Your code, and it turns Your code into a tree-like representation of Your code, in which each node represents meaningful components. Those meaningful components are displayed here. The tree-like representation is stored in the form of the Swift „Abstract Syntax Tree“ (Swift AST), and written in the form of an XML file.

Scope: (Statements)

Blocks: { Statements }

Subscripts: [Number]

Compiler Directives: @

Statements

Simple Statements

Declarations

A declaration introduces a new name or construct into your program.

- Code Block
- Top-Level Code
- Import Declaration
- Constant Declaration
- Variable Declaration
  - Stored Variables & Properties
  - Computed Variables & Properties
  - Mutable Values
  - Property Observers
  - Static Variable Properties
  - Variables in Protocol Definitions
- Type Alias Declaration

Expressions

- Prefix Expressions
  - with try Operator
  - Basic Operators
- Advanced Operators
- Binary Expressions
  - with Assignment Operator
  - with Ternary Conditional Operator
  - with Type Casting Operator
    - is
    - as
    - as?
    - as!
- Primary Expressions
  - syntactically, all Primary Expressions are also Postfix Expressions
  - Literal Expressions
    - Ordinary Literal
      - String Literal
      - Number Literal
        - Int
        - Float
        - Double
    - Dictionary Literal
    - Array Literal
    - Playground Literal
    - Special Literals
      - #file
      - #line
      - #column
      - #function
  - Self Expressions
  - Superclass Expression
  - Closure Expression
  - Capture Lists
  - Implicit Member Expression
  - Paranthesized Expression
  - Tuple Expression
  - Wildcard Expression
  - Expressions for Interoperability with C & Objective C
    - Key Path Expression
    - Selector Expression
    - Key Path String Expression
- Postfix Expressions
  - Function Call
  - Initializer
  - Postfix Self Expression
  - Explicit Member Expression
  - Forced Value Expression
  - Subscript Expression
  - Optional Chaining Expression

Control Flow Statements

Loop Statements

- for-in
- while
- repeat-while
- Alterations
  - break
  - continue

Branch Statements

- if
  - else
  - else if
- guard
- switch
- Alterations
  - break
  - continue

Conditions

- Bool
- Type Bridged to Bool

Control Transfer Statements

- break
- continue
- fallthrough
- return
- throw

Defer

Do

- introduces new scope
- catch

Compiler Control Statements

Conditional Compilation Block

Line Control Statement