LearnAppMaking.com

Swift 3 Cheatsheet

Variables

```
var meaningOfLife:Int = 42
let pi:Double = 3.14159265359 // Constant
var phi:Float = 1.618
var message:String = "Welcome"
var isCurrentUser:Bool = false
var three:Int = 1 + 2 // Expression
var foo = "bar" // Inferred
var optionalMessage:String? // Optional

let a = 3
let b = 60
let c = 27

var isActive = false
```

Functions

```
func greetUser(name: String, bySaying
greeting:String = "Hello") -> String
{
    return "\(greeting), \(name)"
}

let message = greetUser(name: "Reinder",
bySaying: "Good Morning")

print(message)
// Outputs: Good Morning, Reinder
```

Classes

```
class Office: Building, Constructable
{
    var address:String = "1 Infinite Loop"
    var phone:String? // Optional

    @IBOutlet weak var submit:UIButton?

    lazy var foo:String = { // Lazy Property
        return "bar";
    }()

    override init()
    {
        address = "1 Probability Drive"
    }

    func startWorking(_ time:String,
withWorkers workers:Int)
    {
        print("Starting working at time
        \(time) with workers \(workers)")
    }
}
```

Instances

```
var headquarters:Office = Office();
headquarters.phone = "123-456-789"
headquarters.startWorking("09:00",
  withWorkers: 19)
```

Code is art. (Unknown)

CLICK TO TWEET

Control Flow

```
if isActive
{
    print("This user is active.")
} else {
    print("This user is **inactive**")
}

var user:String = "Bob"

if user == "Alice" && isActive
{
    print("Alice is active.")
}
else if user == "Bob" && !isActive
{
    print("Bob is lazy.")
}

if user == "Deep Thought" || meaningOfLife == 42
{
    print("It's either Deep Thought, or
    the meaning of life is 42...")
}

if c < a && b + c == a
{
    print("This will never happen!")
}

for i in 1...5
{
    print(i); // 1 2 3 4 5
}

switch a {
    case 1:
        print("abra")
    case 3:
        print("cadabra")
    default:
        println(“abracadabra")
}

while b <= 60 && b > 0
{
    print(b)
    b -= 1
}
```

LearnAppMaking.com

Strings

```
var me:String = "App Maker"
var ceo:String = "CEO"
var title = "\(me), \(ceo)"

var amount:String = "1234"
var num:Int? = Int(amount) // Optional
```

Share The Love

Check out this neat Swift Cheatsheet for making iPhone apps: http://bit.ly/2diz7Um via @reinderdevries

CLICK TO TWEET

Optionals

```
var bill = "133.70"
var optionalNumber:Double? = Double(bill)

// Optional binding
if let definiteNumber = optionalNumber
{
    print(definiteNumber)
}

// Force-unwrapping
if(optionalNumber != nil)
{
    print(optionalNumber!)
}
```

Special / Intermediate

```
// Optional chaining
cell?.label?.text = "Hello World!"

// Nil-coalescing operator: default value if nil
var meaningOfLife:Int = deepThought.think() ?? 42;

// Downcasting with optional binding
if let book = item as? Book
{

}

// Force downcasting
var book = item as! Book
```

Dictionaries

```
var futurama:[String: String] = [
    "Delivery Boy": "Fry",
    "Robot":        "Bender",
    "Driver":       "Leela",
    "Why Not":      "Zoidberg?",
    "Idiot":        "Zapp Brannigan"
]

for (job, name) in futurama
{
    print("\(name), \(job)");
}

var fry = futurama["Delivery Boy"]
```

Arrays

```
var hitchhikers:[String] = ["Ford",
"Arthur", "Zaphod", "Trillian", "Marvin",
"Slartibartfast"]

hitchhikers += ["Deep Thought"]

var humma:String = "Humma Kavula"
hitchhikers += [humma]

var zaphod:String = hitchhikers[2] // Not 3

for character:String in hitchhikers {
    print(character)
}
```

LearnAppMaking.com

Closures

```
var closure = {
 (name:String,  clearance:Int) -> Bool in

    return name == "Reinder"
        && clearance > 7
}

let auth = closure("Reinder", 9)

func doTask(completionHandler: () -> Void)
{
    // Do the task, then call completionHandler
}

doTask(completionHandler: {
    // Do this when task finishes
})

doTask {
    // Identical as above!
}
```

Generics

```
func generateClones<T>(of clone: T, howMany
n: Int) -> [T]
{
    var cloneContainer : [T] = []

    for _ in 0 ..< n
    {
        cloneContainer.append(clone)
    }

    return cloneContainer
}

let clones = generateClones(of: "Agent
Smith", howMany: 100)
```

Guard & Defer

```
func isMersenne(_ a: Int) -> Bool
{
    guard a > 0 else {
        return false
    }

    // Do the magic

    return true
}

let prime = isMersenne(127)

func createLog()
{
    let file = fopen()

    defer {
        fclose(file)
    }

    let statusA = getInputStatus()

    guard statusA != "error" else
    {
        return
    }

    file.write(statusA)

    let statusB = getOutputStatus()

    guard statusB != "interrupted" else
    {
        return
    }

    file.write(statusB)
}
```

Tuples

```
let coffee = ("Frappuccino", 5.99)

let (coffeeType, coffeePrice) = coffee
print(coffeeType) // Frappuccino

let (_, justThePrice) = coffee
print(justThePrice) // 5.99

let flight = (code: "XJ601", heading:
"North", passengers: 216)
print(flight.heading) // North
```

Error Handling

```
enum MonkeyError: Error {
    case angry
    case tired
    case sad
}

func monkeyMadness() throws
{
    throw MonkeyError.angry
}

do {
    try monkeyMadness()
}
catch let error {
    print(error)
}
```

Further reading: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html