Cheatography

Swift 4 Syntax (beginner) Cheat Sheet

by [deleted] via cheatography.com/66851/cs/16721/

If / else if / else

```
if var1 == var2 {
        pri nt( " Ok")
} else if var1 < var2 {
        pri nt( "Not ok")
} else {
        pri nt( " Maybe ok")
}
```

Switch

```
switch (var1, var2) {
    case (3, 8):
        print("Su re")
    case (1...15, 20..<25):
        print("Ra nge s")
    case (let localVar1, let localVar2) where
localVar1 + localVar2 > 30:
        print(local Var1, localVar2)
    default:
        print("No thi ng")
}
```

while / repeat while

```
var digit = 5
while digit > 0 {
        digit -= 1
        pri nt( digit)
}
repeat {
        digit -= 1
        pri nt( digit)
} while digit >= 0
```

for in

```
for item in ["item1", "item2"] {
        pri nt( item)
}
```

Guard

```
guard var1 >= 3 else {
      pri nt( " Lower than 2")
      con tinue
}
```

Optionals

```
var optional1: String?
var optional2: Int? = 10
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Basic class

```
class Adventurer {
       var name: String
       var hp: Int
       let maxHealth: Int = 100

        static var credo = " Defend the helple ss"

       var specia lMove: String?

       var toFullHp: Int {
                return maxHealth - hp
       }

       var health: Int {
               get {
                        return hp
               }
               set {
                       hp = newValue
               }
       }

        ini t(name: String, hp: Int) {
                sel f.name = name
                self.hp = hp
       }

        con ven ience init(name: String) {
                sel f.i nit (name: name, hp: 100)
       }
}
```

Subclass

```
class Adventurer {
       var name: String
        static var credo = " Defend the helple ss"

        ini t(name: String) {
                sel f.name = name
       }

       func attack (da mage: Int) {
                pri nt( " Att acking for \(damage)
damage.")
       }

        static func printC redo() {
                pri nt( credo)
       }
}
class Ranger: Adventurer {

       var classA dva ntage: String

        ini t(name: String, advantage: String) {
                sel f.c las sAd vantage = advantage
                sup er.i ni t(name: name)
       }

        ove rride func attack (da mage: Int) {
                pri nt( " Ranger attack for \
(dama ge) ")
       }
}
var adventurer = Advent ure r(name: " Har ris on")
var ranger = Ranger (name: " Ste ven ", advantage:
" Ste alt h")
```


Sponsored by ApolloPad.com
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Array

```
var Array: Array<String> = []
var Array2 = Array< Str ing >()
var Array3 = [Double]()
var Array4: [Int] = []
```

Dictionary

```
var dict1: Dictionary<Int, Int> = [:]
var dict2 = Dictio nar y<Int, String >()
var dict3 = [String : String]()
var dict4: [String : String] = [:]
```

Tuple

```
var tuple = (string: "str", int: 1, isBool: true)
var tuple2: (string: String, int: Int, isBool:
Bool)
tuple2 = ("st r", 1, true)
var tuple3: (String, Int, Bool) = ("st r", 1,
true)
```

Set

```
var set = Set<Int>()
var set2: Set<In t> = []
```

Basic function

```
func myFunction(value: String) -> Bool {
        pri nt( " Basic functi on")
        return true
}
```

Complex function

```
func myFunction(value1: String = "Value1", value2:
Int = 2) -> (return1: String, return2: Int) {
      pri nt( " Complex functi on")
      return ("st rin g", 3)
}
```

Functions as parameter

```
func superFunc(value: Int) -> Int {
        return value * 4
}
func myFunc (value: Int, anothe rFunc: (Int) ->
Int) {
       let var1 = anothe rFu nc( value)
        pri nt( var1)
}
myFunc(2, superFunc)
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Empty Closure

```
var closureDeclaration: () -> () = {}
```

Basic shorthand

```
var closure = { (parameterString: String) -> Void
in
        pri nt( par ame ter String)
}
```

Closure shorthand

```
var closure: (String) -> String = { value in
        return " \(v alu e)"
}
```

Closures as function parameters

```
func myFunc(closure: ([String]) -> Void) {
       let array = ["va lue 1", " val ue2 ", " -
val ue3 ", " val ue4 "]
        clo sur e(a rray)
}
myFunc { (values) in
       for value in values {
                pri nt( " \(v alu e)")
       }
}
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com