A Swift Kickstart

Daniel H Steinberg
@dimsumthinking

Variables and Constants

Let

```
let person: String
```

Let

```
let person: String
person = "CocoaConf Attendee"
```

Let

```
let person: String = "CocoaConf Attendee"
```

Type Inference

```
let person = "CocoaConf Attendee"
```

No Automatic Promotion

```
let someInt = 6

let someDouble = 5.0

let productAsDouble = someInt * someDouble
```

Create don't Cast

```
let someInt = 6

let someIntAsADouble = Double(someInt)

let someDouble = 5.0

let productAsDouble = someIntAsADouble * someDouble
```

Let => Constant

```
let person = "CocoaConf Attendee"
println(person)

person = "Java Developer"
```

Var => Variable

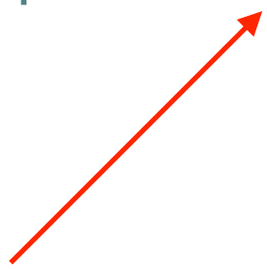
```
let person = "CocoaConf Attendee"

var place: String

place = "Shaker Heights"
```

Favor let over var

```
let person = "CocoaConf Attendee"
var place = "Shaker Heights"
```

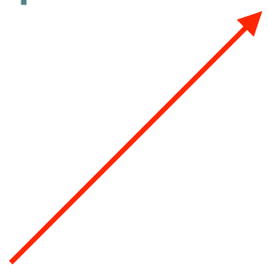
Spaces matter

```
let person = "CocoaConf Attendee"
var place = "Shaker Heights"

let greeting = person + ", welcome to " + place +"."
```



Spaces matter

```
let person = "CocoaConf Attendee"
var place = "Shaker Heights"

let greeting = person + ", welcome to " + place + "."
```



Var

```
let person = "CocoaConf Attendee"

var place = "Shaker Heights"

place = "Las Vegas"

let greeting = person + ", welcome to " + place + "."
```

Types

```
let person = "CocoaConf Attendee"

var place = "Shaker Heights"

place = 7

let greeting = person + ", welcome to " + place + "."
```

Function Results

```
let person = "CocoaConf Attendee"

func columbusWelcome(name: String) -> String {
    return "\(name), welcome to Columbus."
}

let greeting = columbusWelcome(person)
```

Function Results

```
let person = "CocoaConf Attendee"

func columbusWelcome(name: String) -> String {
    return "\(name), welcome to Columbus."
}
```

```
func lasVegasWelcome(name: String) -> String {
    return "\(name), welcome to Las Vegas."
}

let greeting = lasVegasWelcome(person)
```

Traditional Option

```
let person = "CocoaConf Attendee"
var place = "Las Vegas"

func generalWelcome(name: String, location: String) -> String {
    return "\(name), welcome to \(location)."
}

let greeting = generalWelcome(person, place)
```

Functions as variables

```
let person = "CocoaConf Attendee"

func columbusWelcome(name: String) -> String {
    return "\(name), welcome to Columbus."
}

func lasVegasWelcome(name: String) -> String {
    return "\(name), welcome to Las Vegas."
}

var greeting = columbusWelcome

greeting(person)
```

Functions as variables

```
let person = "CocoaConf Attendee"

func columbusWelcome(name: String) -> String {
    return "\(name), welcome to Columbus."
}

func lasVegasWelcome(name: String) -> String {
    return "\(name), welcome to Las Vegas."
}

var greeting = columbusWelcome
greeting = lasVegasWelcome
greeting(person)
```

Functions that return functions

```
func columbusWelcome(name: String) -> String {
    return "\(name), welcome to Columbus."
}
```

```
func lasVegasWelcome(name: String) -> String {
    return "\(name), welcome to Las Vegas."
}

func greetingForLocation(location: String) -> (String) -> String {
    func locationWelcome(name: String) -> String {
        return "\(name), welcome to \(location)"
    }
    return locationWelcome
}
```

Partial Application

```
func greetingForLocation(location: String) -> (String) -> String {
    func locationWelcome(name: String) -> String {
        return "\(name), welcome to \(location)"
    }
    return locationWelcome
}

let columbus = greetingForLocation("Columbus")
let lasVegas = greetingForLocation("Las Vegas")
columbus(person)
```

Functions that accept functions

```
func welcome(personNamed name:String,
    withMessage greeting:(String) -> String) -> String {
        return greeting(name)
}

welcome(personNamed: person, withMessage: columbus)
```

Try this

- Create constants for the coffeeCup and sixteenthNotes
- Revise display() so that theCharacter doesn't have a default value and the character is followed by a \t
- Create displayLine() that takes an int as the second argument and returns a String with that character that many times
- Create displayLines() that takes an int as the second argument and returns a String with the character once in the first row, twice in the second row,… n times in the nth row.

Try this

```
let coffeeCup = "\u{2615}"
let sixteenthNotes = "\u{266C}"

func display(theCharacter: String) -> String {
    return theCharacter + "\t"
}
func display(times: Int, theCharacter: String ) -> String {
    var singleLineDisplay = ""
    for i in 1 ... times {
        singleLineDisplay += display(theCharacter)
    }
    return singleLineDisplay
}

func displayLines(lines: Int, theCharacter: String) -> String {
    var multipleLineDisplay = ""
    for i in 1 ... lines {
        multipleLineDisplay += display(i, theCharacter) + "\n"
    }
    return multipleLineDisplay
}

displayLines(4, coffeeCup)
displayLines(10, sixteenthNotes)
```

https://itunes.apple.com/us/book/a-swift-kickstart/id891801923?mt=11&uo=4&at=11l56E