**Plan for Swift Day**

1. Introduction to Swift/Playgrounds - 10 minutes

Where it came from... problems with Objective-C: C, syntax, legacy

Apple leading up to it - surprise to us, but in the works for a couple of years

Why - safer, faster, modern, interactive

Safer - strong typing (with as little work as possible), other error checking (e.g. inits), handling of pointers.

More powerful - very high ceiling (generics, closures, functional programming)

Plan for day:

[Diagram: Language, Swift libraries, Cocoa libraries]

Morning: features of the language, including Swift libraries.

Afternoon: how it works out with iOS through Cocoa libraries]

Morning through Playgrounds, afternoon through looking at apps.

Playgrounds - what they are, why they are great, examples

[Great place to experiment or to work out details]

*Typing first piece of working code and seeing the values and timeline*

*Using the assistant editor for console output*

*Option-clicking to see declaration of variables*

Explain **s2** - plotting of values inline, timeline of individual values, quicklook.

Show **s3** - picture of what is happening over time as you change things in a ui.

Playgrounds are OS-specific (i.e. you can't run iOS code, e.g. UIKit, in an OSX playground).

Playground settings can be changed (View/Utilities/File Inspector) **S1** or button in top right.

Both platform AND whether documentation is text or interpreted.

- easy to make documented playgrounds for people - just put doc parts in markup.

Still a bit buggy - when nothing happening, worth looking at console in Assistant Editor.

Only other drawback - no user input at present.

**2. Easy bits of language**

Declaration with let and var... use let if you can

*Simple Variables (Int, Double, Float, Boolean, enum)*

(Talk about let age = 22+ 7 and let age= 22+7 not working but with OR without both spaces works ???)

let x = 1

Need strings with values now (String Interpolation)

printing values -> println( "Value is now \(x)")

or in Playground just "Value is now \(x)"

(Using console versus inline reporting)

*Simple constructs - if, for, switch.*

- lack of brackets round statement...

- MUST have curly brackets around then part / else part

Slightly larger example

*Generate how many prime numbers are between 1 and 2500 simply*

[do Erastosthenes later...]

Functions - parameters, returns (tuples!), first class items, strongly typed with a signature, not just as "func".

*Update the above with some simple functions*

Strings, Arrays (and tuples), Dictionaries,

Simple string expressions and printing out

let emptyString = ""

let greeting = "Hello"

let friendlyGreeting = "Hello" + ", friend"

greeting += ", matey"

if emptyString.isEmpty ...

if greeting.hasPrefix( "Hell" )

hasSuffix

== // made up of same characters in same order

< // lexically less

All characters are unicodes

Character handling not well done, changing in Swift 2.

*Another game - Fizz/Buzz*

**3. Classes and Structs**

Classes - initializers, methods, instance variables, properties, computed values

Structs - implicit initializers, pass by value (copy on write), no subclassing - lightweight classes?

Example game ????? Card game????

Extensions...

**4. Generics**

What they are for.

Give example. Won't say more, as if you do that stuff it's obvious, and if you don't, you won't care...

**5. Advanced functions**

Qualified functions?? - making versions of functions with some parameters already supplied

Closures

Functional programming

Optionals?????

[Use string.toInt as easiest example]....

**6. Building Swift Apps**

Hello world.

Two interacting screens (using closures)

Tables

SQL Databases (Bible365)

Reading the documentation.

**7. Finding out more about Swift**

????????????????