Introduction to Swift – Playground

Timothy Chong

Overview

Apple introduced the language Swift during WWDC 2014. XCode 6, packaged with playground, was also introduced.

Playground is a programming tool that provides a fast and simple way for Swift programmers to visualize their codes.

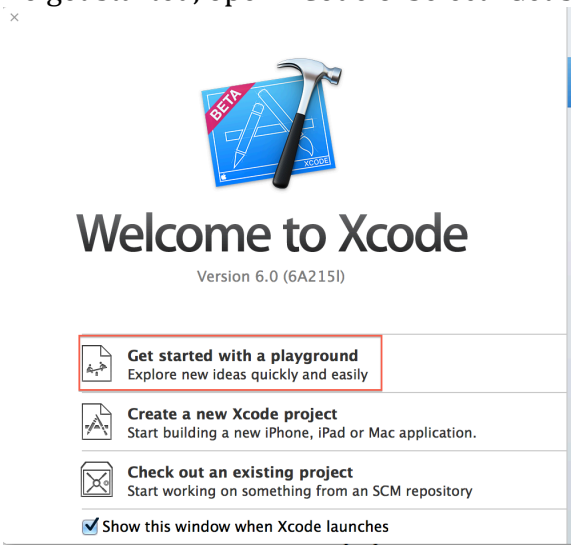
Normally, for all other programming languages, programmers either have to manually compile their codes or use interpreters to translate source-codes to machine executables. With Swift and playground, programmers can observe the behavior of their codes as they type them.

In this tutorial, we will look into the basic functionalities of Playground.

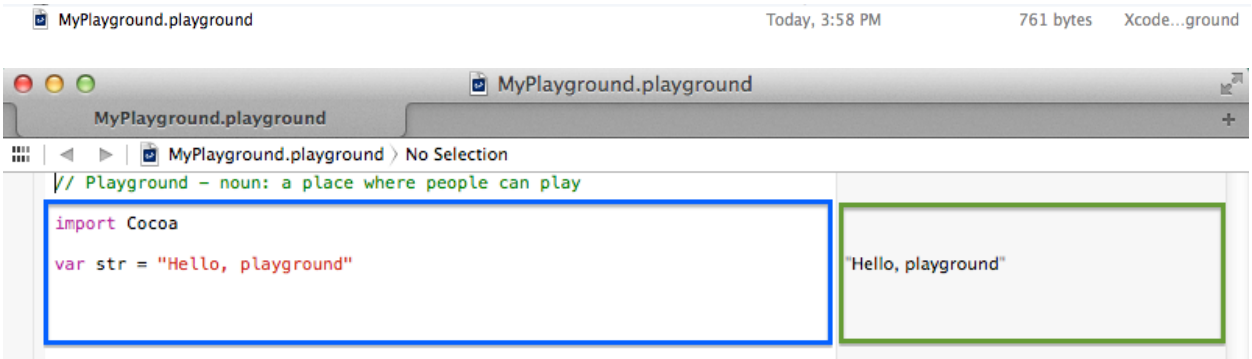
Note: Bear in mind that the purpose of this tutorial is to give you a brief overview of the functionalities of Playground. We will go into more details on codes in subsequent tutorials.

Playground

To get started, open XCode 6. Select “Get started with a playground”



Select the desired location to save your playground file. A file with extension .playground will be created.



The window above will show up. This is the place where you will program in.

The window can be divided into two columns, as shown as the blue and green box in the image above.

The left column is where one types codes in, where as the right column updates automatically according to what is typed in the left column. The right column provides a very convenient way to examine the behavior of the program one writes.

In the example above, in line `var str = “Hello, playground”,` variable with name “str” is initiated with type String and content “Hello, playground”. Essentially, a sets of characters grouped together (a string) is stored in “str”.

The right hand column shows the value stored inside “str”, which in this case is obviously “Hello, playground”.

How can Playground help you?

Playground can give you simple calculation results. In the example below, two variables of names num1 and num2 are made; the sum is set to be the sum of the two. Without actually having to press compile, the right column shows the result of sum.

```
let num1 = 2                    2
let num2 = 3                    3
let sum = num1 + num2           5
```

Of course, it would also work with more complicated situations. A function is basically a grouping of codes that perform some specific operations. Depending on the implementation of a function, it can receive values (inputs) and return values (outputs). In this case the function "Fibonacci" expects a number as its inputs (position of number in the Fibonacci sequence), and returns a number result (the Fibonacci value at the position).

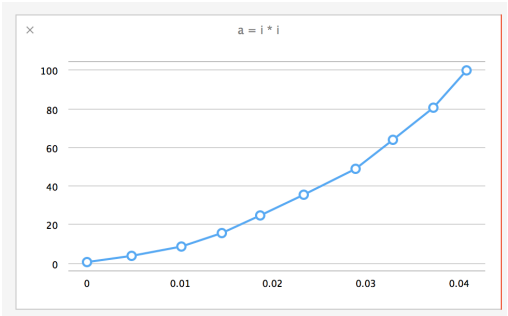
```
func fibonacci (num:Int) -> Int{
    switch(num){
    case 0,1:
        return num                          (54 times)
    default:
        return fibonacci(num - 1) + fibonacci(num - 2)      (46 times)
    }
    //Function doesn't work with negative integers
}

fibonacci(0)                                0
fibonacci(1)                                1
fibonacci(2)                                1
fibonacci(3)                                2
fibonacci(4)                                3
fibonacci(5)                                5
fibonacci(6)                                8
fibonacci(7)                                13
```

With playground, you can easily check if your function implementation is correct. Notice that playground also shows how many times a specific line in a function runs.

If there are some lines of codes that are evaluated multiple times, you can also observe the results of those evaluation in a graphical form. Simply press the + button in the right column (you need to hover over the line for it to show up). The graph will then show up.

In this example, the squares of the values from 1 to 10 are evaluated. The results are clearly shown in the graph to the right.

```
var a = 0                           0
for i in 1...10
{
    a = i * i                       (10 times)
}
```



In some cases, it is very helpful to physically see the elements that you want to put inside an application. Playground lets you do that. The example below shows the creation of a frame. By pressing the eye-looking button in the right column, the visualizing window pops up, which shows you a clear picture of the frame that you created. This can come very handy when one is not sure whether the code that he creates in code is what he has in mind.

```
let frame = NSRect(x: 0, y: 0, width: 40,
    height: 50)
```

{x 0 y 0 w 40 h 50}