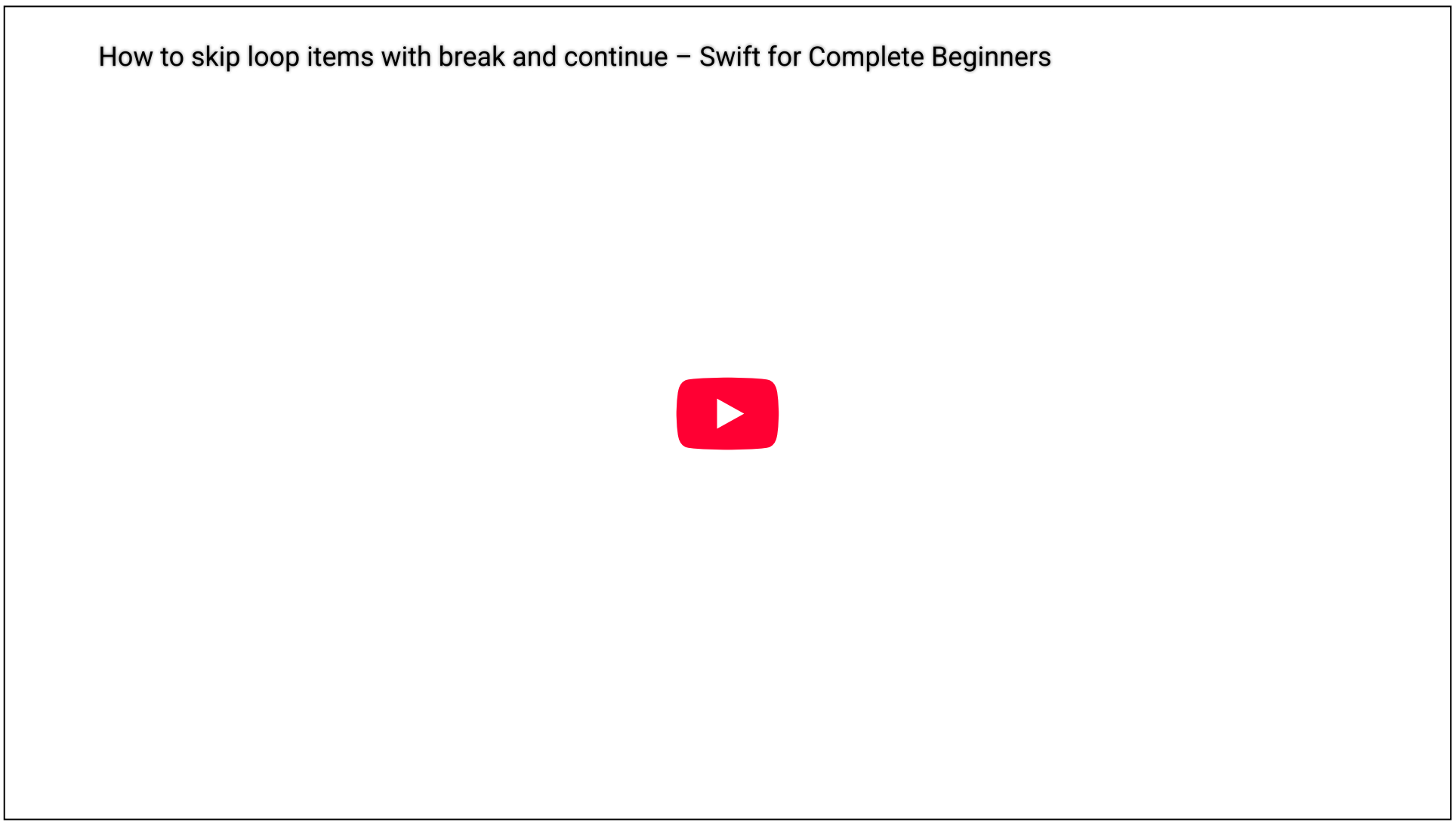
< How to use a while loop to repeat work

Summary: Conditions and loops >

How to skip loop items with break and continue

Paul Hudson @twostraws October 5th 2021

Updated for Xcode 16.4



Swift gives us two ways to skip one or more items in a loop: `continue` skips the current loop iteration, and `break` skips all remaining iterations. Like `while` loops these are *sometimes* used, but in practice much less than you might think.

Let's look at them individually, starting with `continue`. When you're looping over an array of data, Swift will take out one item from the array and execute the loop body using it. If you call `continue` inside that loop body, Swift will immediately stop executing the current loop iteration and jump to the next item in the loop, where it will carry on as normal. This is commonly used near the start of loops, where you eliminate loop variables that don't pass a test of your choosing.

Here's an example:

```
let filenames = ["me.jpg", "work.txt", "sophie.jpg", "logo.psd"]

for filename in filenames {
    if filename.hasSuffix(".jpg") == false {
        continue
    }

    print("Found picture: \(filename)")
}
```

That creates an array of filename strings, then loops over each one and checks to make sure it has the suffix ".jpg" – that it's a picture. **`continue`** is used with all the filenames failing that test, so that the rest of the loop body is skipped.

As for **`break`**, that exits a loop immediately and skips all remaining iterations. To demonstrate this, we could write some code to calculate 10 common multiples for two numbers:

```
let number1 = 4
let number2 = 14
var multiples = [Int]()

for i in 1...100_000 {
    if i.isMultiple(of: number1) && i.isMultiple(of: number2) {
        multiples.append(i)

        if multiples.count == 10 {
            break
        }
    }
}

print(multiples)
```

That does quite a lot:

1. Create two constants to hold two numbers.
2. Create an integer array variable that will store common multiples of our two numbers.
3. Count from 1 through 100,000, assigning each loop variable to **`i`**.
4. If **`i`** is a multiple of both the first and second numbers, append it to the integer array.
5. Once we hit 10 multiples, call **`break`** to exit the loop.
6. Print out the resulting array.

So, use **`continue`** when you want to skip the rest of the current loop iteration, and use **`break`** when you want to skip all remaining loop iterations.

< How to use a while loop to repeat work

Summary: Conditions and loops >

Was this page useful? Let us know!

☆☆☆☆☆

Average rating: 4.6/5

You are not logged in

Log in or create account