Find difference in seconds between NSDates as integer using Swift

Asked 10 years, 1 month ago Modified 1 year, 7 months ago Viewed 97k times

111

I'm writing a piece of code where I want to time how long a button was held down. To do that I recorded an `NSDate()` when the button was pressed, and tried using the `timeIntervalSinceDate` function when the button was released. That seems to work but I can't find any way to print the result or switch it to an integer.

```
var timeAtPress = NSDate()

@IBAction func pressed(sender: AnyObject) {
    println("pressed")
    timeAtPress = NSDate()
}

@IBAction func released(sender: AnyObject) {
    println("released")
    var elapsedTime = NSDate.timeIntervalSinceDate(timeAtPress)
    duration = ???
}
```

I've seen a few similar questions, but I don't know C so I had a hard time understanding the answers given. If there is a more efficient way to find out how long the button was held down I'm open to suggestions.

swift nsdate nstimeinterval

Share
Improve this question
Follow

edited Dec 18, 2022 at 20:55
starball
48.3k ● 28 ● 187 ● 848

asked Oct 28, 2014 at 0:44
Erik
2,399 ● 5 ● 20 ● 24

6 Answers

Sorted by: Highest score (default)

248

Your attempt to calculate `elapsedTime` is incorrect. In Swift 3, it would be:

```
let elapsed = Date().timeIntervalSince(timeAtPress)
```

Note the `()` after the `Date` reference, which calls `Date.init`.

Alternatively, nowadays (e.g., iOS 15+, macOS 12+), we might prefer to use `now`, which does precisely the same thing, but makes the functional intent explicit through its name:

```
let elapsed = Date.now.timeIntervalSince(timeAtPress)
```

The `Date()`/`Date.init()`/`Date.now` instantiates a new date object, and then `timeIntervalSince` returns the time difference between that and `timeAtPress`. That will return a floating point value (technically, a `TimeInterval`).

If you want that as truncated to a `Int` value, you can just use:

```
let duration = Int(elapsed)
```

There are a few alternatives:

1. Nowadays (iOS 16+, macOS 13+), we might use a `Clock`, e.g., a `ContinuousClock` or a `SuspendingClock`:

   ```
   let clock = ContinuousClock()
   let start = clock.now

   // do something

   let elapsed = .now - start
   ```

   Or we can `measure` the amount of time something takes:

   ```
   // for async routines

   let elapsed = try await ContinuousClock().measure {
       // something asynchronous with `await`
   }

   // for synchronous routines

   let elapsed = ContinuousClock().measure {
       // something synchronous
   }
   ```

   These clocks are introduced about 6 minutes into WWDC 2022 video Meet Swift Async Algorithms.

2. Sometimes, we just want the number of elapsed seconds, e.g., with `CFAbsoluteTimeGetCurrent()`:

   ```
   let start = CFAbsoluteTimeGetCurrent()

   // do something
   ```

```
let elapsed = CFAbsoluteTimeGetCurrent() - start
```

3. It's worth noting that the `CFAbsoluteTimeGetCurrent` documentation warns us:

   > Repeated calls to this function do not guarantee monotonically increasing results. The system time may decrease due to synchronization with external time references or due to an explicit user change of the clock.

   This means that if you're unfortunate enough to measure elapsed time when one of these adjustments take place, you can end up with incorrect elapsed time calculation. This is true for `NSDate` / `Date` calculations too. It's safest to use a `mach_absolute_time` based calculation (most easily done with `CACurrentMediaTime`):

   ```
   let start = CACurrentMediaTime()

   // do something

   let elapsed = CACurrentMediaTime() - start
   ```

   This uses `mach_absolute_time`, but avoids some of its complexities outlined in Technical Q&A QA1398.

   Remember, though, that `CACurrentMediaTime` / `mach_absolute_time` will be reset when the device is rebooted. So, bottom line, if you need accurate elapsed time calculations while an app is running, use `CACurrentMediaTime`. But if you're going to save this start time in persistent storage which you might recall when the app is restarted at some future date, then you have to use `Date` or `CFAbsoluteTimeGetCurrent`, and just live with any inaccuracies that may entail.

Share
Improve this answer
Follow

edited May 5, 2023 at 18:07

answered Oct 28, 2014 at 1:11
Rob
436k ● 74 ● 832 ● 1.1k

what's the dimension for timeIntervalSinceDate? – eugene Nov 10, 2015 at 10:55

1 It returns a `TimeInterval`, which is measured in seconds. – Rob Mar 22, 2020 at 17:55

35

Swift 5

```
let differenceInSeconds = Int(endDate.timeIntervalSince(startDate))
```

Share Improve this answer Follow

answered Apr 30, 2020 at 19:14

iKK
6,992 ● 12 ● 68 ● 152

Helpful tip: removing the `Int(..)` wrapper will show decimals of a second instead of rounding to nearest second. – joshuakcockrell Aug 24 at 17:49

24

NSDate() and NSCalendar() sound like a good choice. Use calendar calculation and leave the actual math part to the framework. Here is a quick example of getting the seconds between two `NSDate()`

```
let startDate = NSDate()
let endDate = NSDate()
let calendar = NSCalendar.currentCalendar()
let dateComponents = calendar.components(NSCalendarUnit.CalendarUnitSecond,
fromDate: startDate, toDate: endDate, options: nil)
let seconds = dateComponents.second
println("Seconds: \(seconds)")
```

Share
Improve this answer
Follow

edited Nov 22, 2017 at 9:16
Yassine ElBadaoui
450 ● 7 ● 17

answered Oct 28, 2014 at 2:03
Freddy
2,279 ● 1 ● 22 ● 32

1 For some reason this always returns 0 for me even though the dates are an hour apart. – Markymark Jul 30, 2020 at 4:08

11

According with the Freddy's answer, this is the function in swift 3:

```
let start = Date()
let end = Date(timeIntervalSince1970: 100)
let calendar = Calendar.current
let unitFlags = Set<Calendar.Component>([ .second])
let datecomponents = calendar.dateComponents(unitFlags, from: start, to: end)
let seconds = datecomponents.second
print(String(describing: seconds))
```

Share
Improve this answer
Follow

edited Nov 17, 2019 at 19:58
DoesData
7,029 ● 4 ● 44 ● 64

answered Aug 2, 2017 at 16:00
Andres Paladines
1,218 ● 16 ● 21

0

This is how you can get the difference in latest version of Swift 3

```
let calendar = NSCalendar.current
var compos:Set<Calendar.Component> = Set<Calendar.Component>()
```

```
compos.insert(.second)
compos.insert(.minute)
let difference = calendar.dateComponents(compos, from: fromDate, to: toDate)
print("diff in minute=\(difference.minute!)") // difference in minute
print("diff in seconds=\(difference.second!)") // difference in seconds
```

Reference: [Getting the difference between two NSDates in (months/days/hours/minutes/seconds)]()

Share Improve this answer Follow

answered Jul 29, 2017 at 22:38
Rujoota Shah
1,321 ● 16 ● 18

-6

For Swift 3 Seconds between 2 time in "hh:mm"

```
func secondsIn(_ str: String)->Int{
    var strArr = str.characters.split{$0 == ":"}.map(String.init)
    var sec = Int(strArr[0])! * 3600
    var sec1 = Int(strArr[1])! * 36
    print("sec")
    print(sec+sec1)
    return sec+sec1

}
```

Usage

```
var sec1 = secondsIn(shuttleTime)
var sec2 = secondsIn(dateToString(Date()))
print(sec1-sec2)
```

Share Improve this answer Follow

answered Feb 22, 2017 at 9:27
Rishabh Dugar
606 ● 5 ● 16

This answer is irrelevant to the answer of the op. – Tomasz Nazarenko Apr 9, 2017 at 7:04