PG5600 iOS programming

Lecture 7

Last time

- View concepts
- Instantiating views
- Custom views
- Events
- Gestures
- Animaions

Agenda

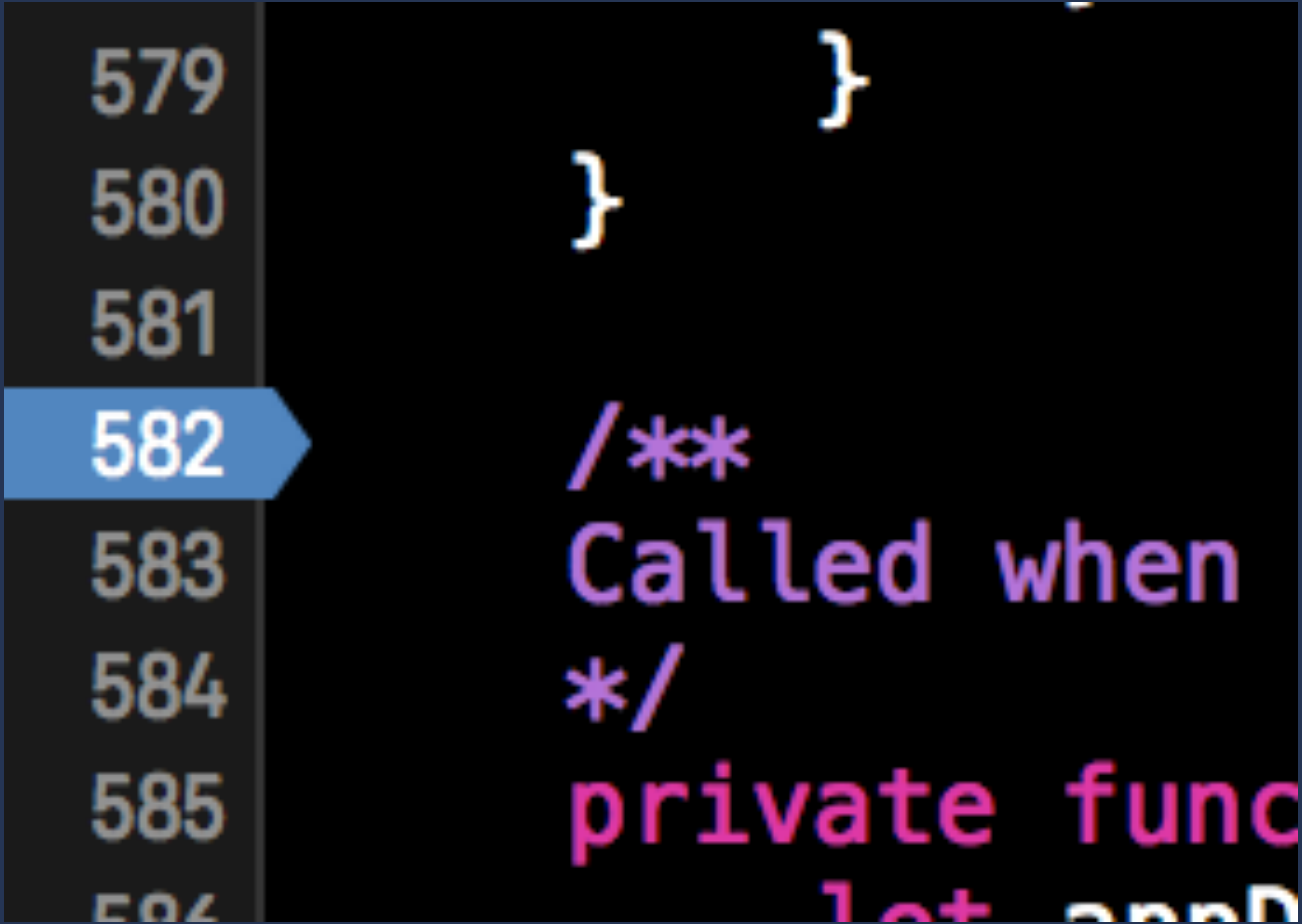
- Debugging
- Testing
- Swift and code reuse
    - Framework
    - Cocoapods & Carthage
- Threads and asyncronicity
- web requests

Debugging

- Breakpoints
- Loggin
- Unit tests
- Assertions (or force unwraps)

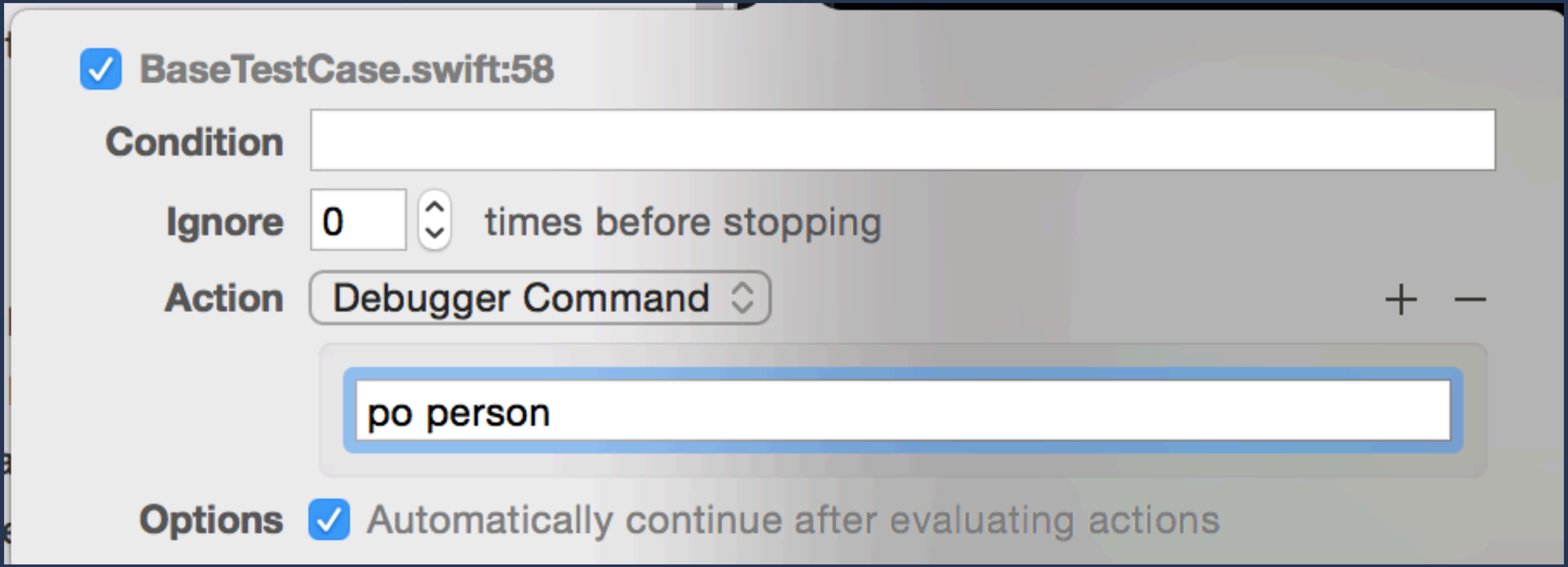
Debugging

Breakpoint Logging



```
        }
    }

    /**
    Called when
    */
    private func
```


☑ BaseTestCase.swift:58

Condition

Ignore 0 times before stopping

Action Debugger Command

```
po person
```

Options ☑ Automatically continue after evaluating actions

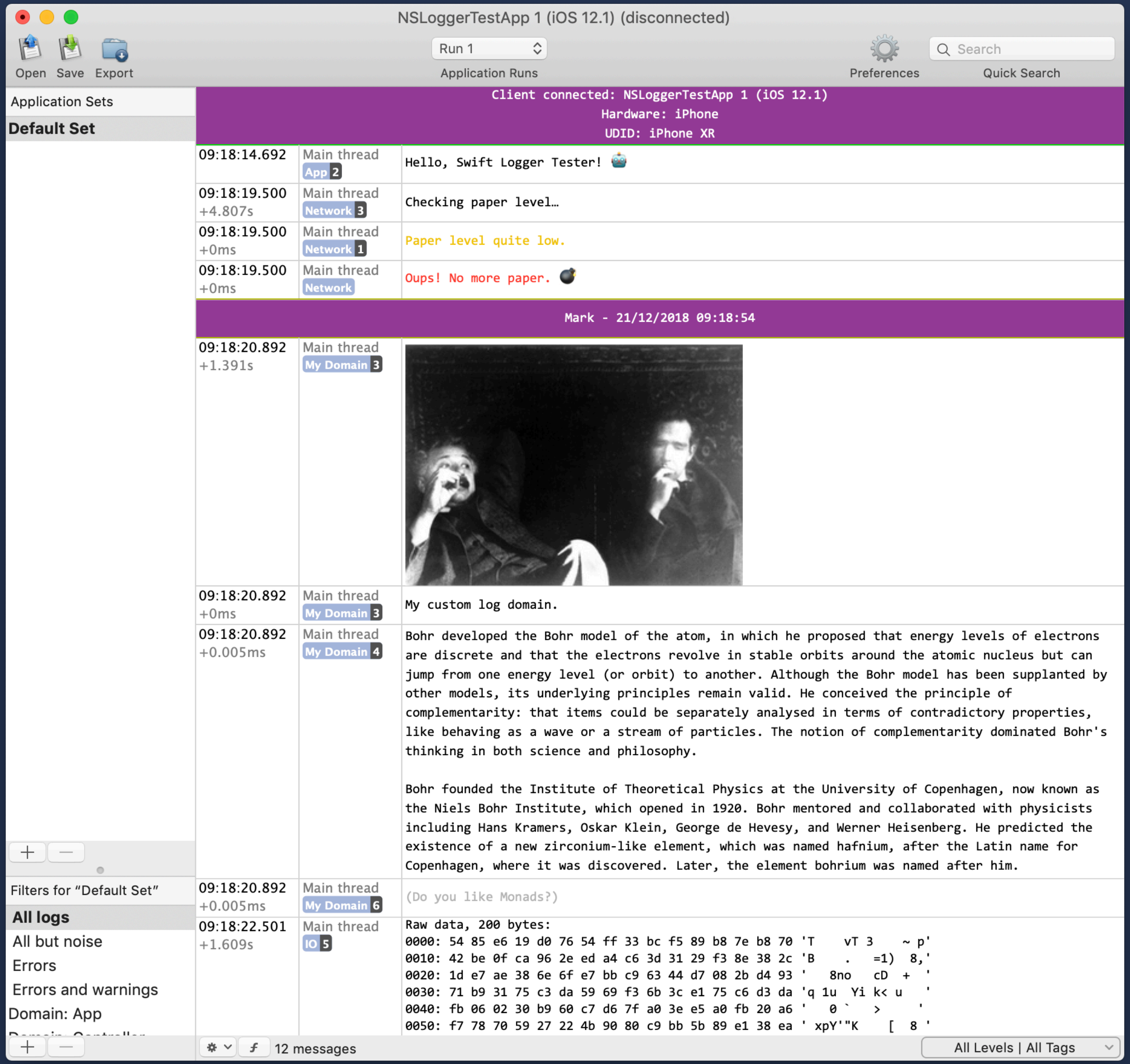
Logging

```
// console i XCode
print("Logg en linje")

// Device console (can be shown in console of device, without xcode)
NSLog("Logg objekter")
```

NSLogger

https://github.com/fpillet/NSLogger



```
Client connected: NSLoggerTestApp 1 (iOS 12.1)
Hardware: iPhone
UDID: iPhone XR

09:18:14.692           Main thread   Hello, Swift Logger Tester!
                       App 2
09:18:19.500 +4.807s   Main thread   Checking paper level…
                       Network 3
09:18:19.500 +0ms      Main thread   Paper level quite low.
                       Network 1
09:18:19.500 +0ms      Main thread   Oups! No more paper.
                       Network

Mark - 21/12/2018 09:18:54

09:18:20.892 +1.391s   Main thread
                       My Domain 3
09:18:20.892 +0ms      Main thread   My custom log domain.
                       My Domain 3
09:18:20.892 +0.005ms  Main thread   Bohr developed the Bohr model of the atom, in which he proposed that energy levels of electrons
                       My Domain 4   are discrete and that the electrons revolve in stable orbits around the atomic nucleus but can
                                     jump from one energy level (or orbit) to another. Although the Bohr model has been supplanted by
                                     other models, its underlying principles remain valid. He conceived the principle of
                                     complementarity: that items could be separately analysed in terms of contradictory properties,
                                     like behaving as a wave or a stream of particles. The notion of complementarity dominated Bohr's
                                     thinking in both science and philosophy.

                                     Bohr founded the Institute of Theoretical Physics at the University of Copenhagen, now known as
                                     the Niels Bohr Institute, which opened in 1920. Bohr mentored and collaborated with physicists
                                     including Hans Kramers, Oskar Klein, George de Hevesy, and Werner Heisenberg. He predicted the
                                     existence of a new zirconium-like element, which was named hafnium, after the Latin name for
                                     Copenhagen, where it was discovered. Later, the element bohrium was named after him.
09:18:20.892 +0.005ms  Main thread   (Do you like Monads?)
                       My Domain 6
09:18:22.501 +1.609s   Main thread   Raw data, 200 bytes:
                       IO 5          0000: 54 85 e6 19 d0 76 54 ff 33 bc f5 89 b8 7e b8 70 'T    vT 3    ~ p'
                                     0010: 42 be 0f ca 96 2e ed a4 c6 3d 31 29 f3 8e 38 2c 'B    .   =1)  8,'
                                     0020: 1d e7 ae 38 6e 6f e7 bb c9 63 44 d7 08 2b d4 93 '   8no   cD  +  '
                                     0030: 71 b9 31 75 c3 da 59 69 f3 6b 3c e1 75 c6 d3 da 'q 1u  Yi k< u   '
                                     0040: fb 06 02 30 b9 60 c7 d6 7f a0 3e e5 a0 fb 20 a6 '   0 `   >      '
                                     0050: f7 78 70 59 27 22 4b 90 80 c9 bb 5b 89 e1 38 ea ' xpY'"K    [  8 '
```

```swift
import NSLogger

[…]

// logging some messages
Logger.shared.log(.network, .info, "Checking paper level…")

// logging image
Logger.shared.log(.view, .noise, myPrettyImage)

// logging data
Logger.shared.log(.custom("My Domain"), .noise, someDataObject)
//Swell.plist for å konfigurere
```

Unit tests

```swift
import XCTest
import SwiftFonts

class FontSorterTests: XCTestCase {

    let sorter = FontSorter()

    func testCompareHyphenWithNoHyphen() {
        let fonts = ["Arial-ItalicMT", "ArialMT"]
        let expected = ["ArialMT", "Arial-ItalicMT"]
        let sorted = sorter.sortFontNames(fonts)
        XCTAssertEqual(expected[0], sorted[0], "the array should be sorted properly")
        XCTAssertEqual(expected[1], sorted[1], "the array should be sorted properly")
    }

    func testCompareHyphenWithHyphen() {
        let fonts = ["Avenir-Roman", "Avenir-Oblique"]
        let expected = ["Avenir-Oblique", "Avenir-Roman"]
        let sorted = sorter.sortFontNames(fonts)
        XCTAssertEqual(expected[0], sorted[0], "when two fonts contain a hyphen, they should be sorted alphabetically")
        XCTAssertEqual(expected[1], sorted[1], "when two fonts contain a hyphen, they should be sorted alphabetically")
    }
}
```

XCode test assertions

```
XCTAssert(expression, format...) // hvis expression = true, så er testen ok

XCTAssertTrue(expression, format...) // lik som den over

XCTAssertFalse(expression, format...) // hvis false så er testen ok

XCTAssertEqual(expression1, expression2, format...) // lik så er testen ok

XCTAssertNotEqual(expression1, expression2, format...) // ulike så er testen ok

XCTAssertEqualWithAccuracy(expression1, expression2, accuracy, format...) // kan brukes på nummer som ikke må være helt lik

XCTAssertNotEqualWithAccuracy(expression1, expression2, accuracy, format...) // kan brukes på nummer som ikke må være helt lik

CTAssertNil(expression, format...) // teste optionals

XCTAssertNotNil(expression, format...) // teste optionals
```

Async testing

```swift
func testAsynchronousURLConnection() {

    let URL = "http://mobile-course.herokuapp.com/message"
    let expectation = expectationWithDescription("GET \(URL)")

    let session = NSURLSession.sharedSession()
    let task = session.dataTaskWithURL(NSURL(string: URL), completionHandler: {(data, response, error) in
        expectation.fulfill()

        XCTAssertNotNil(data, "data should not be nil")
        XCTAssertNil(error, "error should be nil")

        if let HTTPResponse = response as? NSHTTPURLResponse {
            XCTAssertEqual(HTTPResponse.URL!.absoluteString!, URL, "HTTP response URL should be equal to original URL")
            XCTAssertEqual(HTTPResponse.statusCode, 200, "HTTP response status code should be 200")
            XCTAssertEqual(HTTPResponse.MIMEType as String!, "application/json", "HTTP response content type should be text/html")
        } else {
            XCTFail("Response was not NSHTTPURLResponse")
        }
    })

    task.resume()

    waitForExpectationsWithTimeout(task.originalRequest.timeoutInterval, handler: { error in
        task.cancel()
    })
}
```

Performance testing

```
func testPerformanceExample() {
    // Tester performance med self.measureBlock
    self.measureBlock() {
        // Time the stuff here
    }
}
```

Code level assertions

- Optionals for values that *have* to be present for the app to run
- Crashing the app is better than user getting stuck
- Remember to run through the app before release though!

Force unwraps

```
@IBOutlet weak var UILabel titleLabel!
```

Normal assertion (not widely used)

```
assert(age > 13, "Registered age is not above 13")
```

NB remember this common mistake!

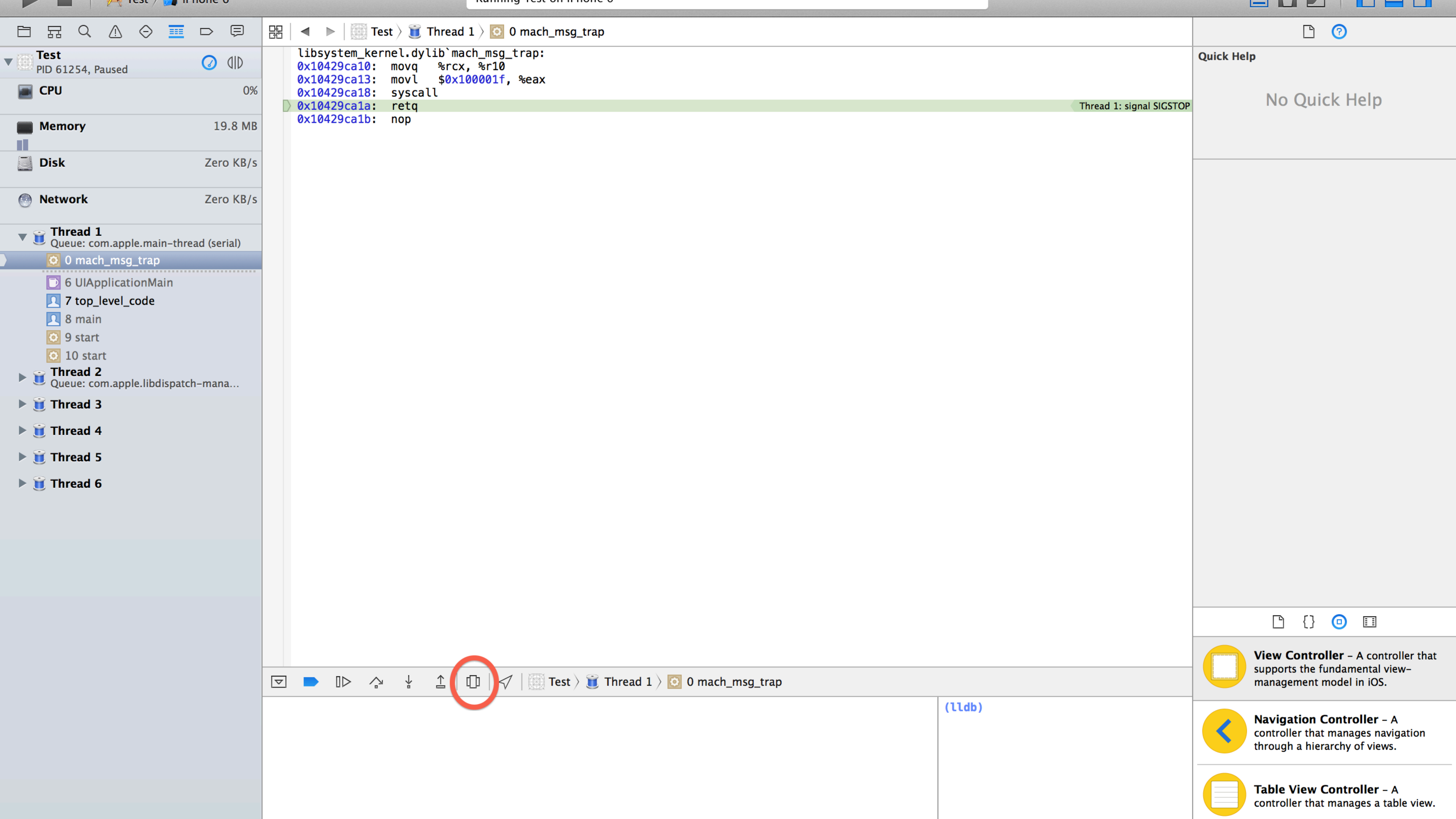
```swift
let myArray = ["aString", "anotherString"]

let value = myArray[3]
```

💥💥💥💥💥💥💥💥💥💥💥

Debugging View

Pause the app while running



```
libsystem_kernel.dylib`mach_msg_trap:
0x10429ca10:  movq   %rcx, %r10
0x10429ca13:  movl   $0x100001f, %eax
0x10429ca18:  syscall
0x10429ca1a:  retq                                        Thread 1: signal SIGSTOP
0x10429ca1b:  nop
```

```
(lldb)
```

Test 〉 Thread 1 〉 0 mach_msg_trap
Test
PID 61254, Paused
CPU 0%
Memory 19.8 MB
Disk Zero KB/s
Network Zero KB/s
Thread 1
Queue: com.apple.main-thread (serial)
0 mach_msg_trap
6 UIApplicationMain
7 top_level_code
8 main
9 start
10 start
Thread 2
Queue: com.apple.libdispatch-mana...
Thread 3
Thread 4
Thread 5
Thread 6
Quick Help
No Quick Help
View Controller – A controller that supports the fundamental view-management model in iOS.
Navigation Controller – A controller that manages navigation through a hierarchy of views.
Table View Controller – A controller that manages a table view.

Debugging

Playground

Debug small functions or views

Code reuse / frameworks

- Import files directly
- Cocoapods
- Carthage
- Swift Package manager

Frameworks

- Makes it easier to reuse code
- Good for an app with many targets
- Remember public / private / internal

Async

- Main thread can draw GUI
- Use other threads for big calculations
- Different options for making threads
  - NSThread
  - Grand Central Dispatch
  - NSOperationQueue

Grand Central Dispatch

- Creates the threads for you
- Based on queues of tasks
- Two types of tasks
    1. Serial - En oppgave av gangen
    2. Concurrent - Kan utføre flere oppgaver samtidig

```
 DispatchQueue.init(label: "another thread").async {
   /// Do stuff

   // Back to main thread!
   DispatchQueue.main.async {
     // draw images, update GUI
   }

}
```

NSTread

- Not widely used

```
// Lag en ny tråd
NSThread.detachNewThreadSelector("someMethod", toTarget: self, withObject: nil)

var thread = NSThread(target: self, selector: "testMethod", object: nil)
thread.start()
thread.cancel()
```

NSOperation, NSOperationQueue

- One unit of work
- Abstract class you inherit from

Alternativer
NSBlockOperation - Create a closure that runs in a thread
NSInvocationOperation - Runs a function in a thread

Starting a NSOperation

```
var operation = NSOperation()
operation.start()

operation.cancel()
```

Put in NSOperationQueue

- Runs a set of NSOperation, NSBlockOperation or NSInvocationOperation
- First-In-First-Out as a standard
- Set max concurrent tasks with `maxConcurrentOperationCount`
- Uses Grand Central Dispatch

- `QOS_CLASS_USER_INTERACTIVE` = NSQualityOfServiceUserInteractive
- `QOS_CLASS_USER_INITIATED` = NSQualityOfServiceUserInitiated
- `QOS_CLASS_UTILITY` = NSQualityOfServiceUtility
- `QOS_CLASS_BACKGROUND` = NSQualityOfServiceBackground

```
let backgroundOperation = NSOperation()
backgroundOperation.qualityOfService = .Background

let operationQueue = NSOperationQueue()
operationQueue.addOperation(backgroundOperation)
```

NSThread vs GCD vs NSOperationQueue

- GCD for easy use, day to day usage
- NSTread if you need full control
- NSOperationQueue if you need to set queues with maxs tasks, specific ordering etc

Web requests

Http methods

GET - Get data

POST - Send data

PUT - Update data

PATCH - Update some fields only

DELETE - Delete data

```swift
let url = NSURL(string: "http://ip.jsontest.com")
let session = NSURLSession.sharedSession()
let task = session.dataTaskWithURL(url, completionHandler: { (data, response, error) -> Void in
    print(data)
})

task.resume()

let url2 = NSURL(string: "http://mobile-course.herokuapp.com/message")
let request = NSMutableURLRequest(URL: url2)
request.HTTPMethod = "POST"
let session2 = NSURLSession.sharedSession()
let task2 = session.dataTaskWithRequest(request, completionHandler: { (data, response, error) -> Void in
    print(data)
})

task2.resume()
```

Playground og Nettverk

For å kjøre asynkron kode i playground må man gjøre følgende

```
import XCPlayground
XCPSetExecutionShouldContinueIndefinitely()
```

Alamofire og REST

https://github.com/Alamofire/Alamofire

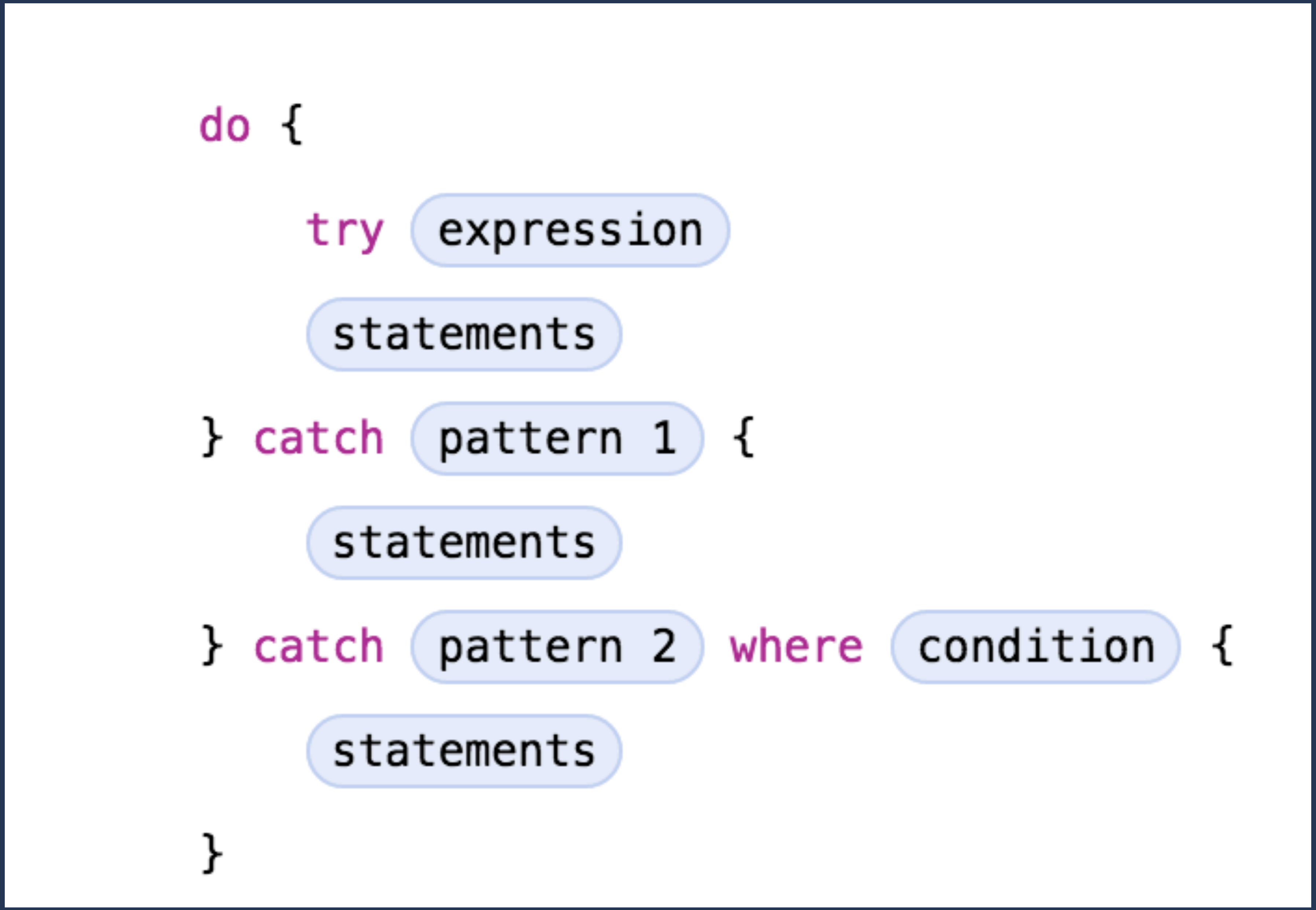
```
Alamofire.request(.GET,  "http://jsonplaceholder.typicode.com/posts")
            .responseJSON { ( response) -> Void in

                if let responseJSONArray = response.result.value as? [[String : AnyObject]] {

                    for post in responseJSONArray {
                        print(post)
                    }
                }

                if let responseError = response.result.error {
                    print(response.result.error)
                }
        }
```

try



```
do {
    try expression
    statements
} catch pattern 1 {
    statements
} catch pattern 2 where condition {
    statements
}
```

```swift
func testStuff() {
    do {
        try login()
    } catch LoginError.NoUserName {
        print("wrong username")
    } catch LoginError.WrongPassword {
        print("wrong password")
    } catch {
        print("some other error")
    }
}
```

Try

```swift
enum LoginError: ErrorType {
    case NoUserName
    case WrongPassword
}

func login() throws {
    defer {
        print("an error happened")
    }

    let userText : String? = "John Snow"
    let passWordIsCorrect = false

    guard let actualUserName = userText else {
        throw LoginError.NoUserName
    }

    guard passWordIsCorrect else {
        throw LoginError.WrongPassword
    }
}
```

Force Try!

```
// crash if error
try! login()
```

JSON
Swift 3

```swift
func titlesFromJSON(data: NSData) -> [String] {
    var titles = [String]()

 do {
    let jsonDictionary = try NSJSONSerialization.JSONObjectWithData(data, options: nil, error: &jsonError) as? [String : AnyObject] {
        guard let feed = jsonDictionary["feed"] as? [String : AnyObject] else {
            // throw ?
        }

    } catch let error as NSError {
        // Do something with error
    }

    return titles
}
```

Find name

```
struct Movie {
  let name: String
  let genre: String?
  let year : Int
  let rating: Double

  init?(attributes: [String : Any]) {

    guard let name = attributes["name"] as? String, let year = attributes["year"] as? Int, let rating = attributes["rating"] as? Double else {
      return nil
    }
      self.name = name
      self.genre = attributes["genre"] as? String
      self.year = year
      self.rating = rating
  }
}

let jsonDict  = NSJSONSerialization.JSONObjectWithData(data,
  options: NSJSONReadingOptions.MutableContainers, error: nil) as? [String : Any]

Movie(attributes: jsonDict)
```

Swift 4

```swift
struct Movie : Decodable{
    let name: String
    let genre: String?
    let year : Int
    let rating: Double

    enum Keys: String, CodingKey {
        case name
        case genre
        case year
        case rating
    }


    public init(from decoder: Decoder) throws {
        let container = try decoder.container(keyedBy: Keys.self)
        self.name = try container.decode(String.self, forKey: Keys.name)
        self.rating = try container.decode(Double.self, forKey: Keys.rating)
        self.year = try container.decode(Int.self, forKey: Keys.year)
        self.genre = try container.decode(String?.self, forKey: Keys.genre)
    }

}
```

```
let json =
"""
{ "name" : "Matrix",
"genre" : "Sci-fi",
"year" : 2003,
"rating" : 9.8
}
"""

let movie = try JSONDecoder().decode(Movie.self, from: json.data(using: String.Encoding.utf8)!)
```

Videre lesning

- https://github.com/ochococo/Design-Patterns-In-Swift
- http://www.raywenderlich.com/79149/grand-central-dispatch-tutorial-swift-part-1
- Error handling i boka
- Basics i iOS-boka
- Cocoapods.org

Oppgaver

Se GitHub