Swift Pills 💊 #2

Vincent Pradeilles - Worldline

Let’s talk about `@autoclosure`

Say we want to write a function that logs messages

Logging messages

```
func log(_ message: String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    print("[\(file):\(line)] \(function) - \(message)")
}
```

Logging messages

```
func log(_ message: String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    print("[\(file):\(line)] \(function) - \(message)")
}
```

Logging messages

```
func log(_ message: String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    print("[\(file):\(line)] \(function) - \(message)")
}
```

Logging messages

```
func log(_ message: String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    print("[\(file):\(line)] \(function) - \(message)")
}
```

Wait, we don’t want to have logs in production 😅

Logging messages

```
enum Environment {
    case debug
    case store
}
```

Logging messages

```
enum Environment {
    case debug
    case store
}

let environment: Environment = .store
```

Logging messages

```
enum Environment {
    case debug
    case store
}

let environment: Environment = .store

func log(_ message: String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    guard environment != .store else { return }

    print("[\(file):\(line)] \(function) - \(message)")
}
```

Logging messages

```
enum Environment {
    case debug
    case store
}

let environment: Environment = .store

func log(_ message: String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    guard environment != .store else { return }

    print("[\(file):\(line)] \(function) - \(message)")
}
```

Logging messages

```
enum Environment {
    case debug
    case store
}

let environment: Environment = .store

func log(_ message: String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    guard environment != .store else { return }

    print("[\(file):\(line)] \(function) - \(message)")
}
```

Is it really a good solution? 🤔

Logging messages

```
func expensiveCall() -> Int {
    return (0...10_000_000).reduce(0, +)
}
```

Logging messages

```
func expensiveCall() -> Int {
    return (0...10_000_000).reduce(0, +)
}

log("Result was: \(expensiveCall())")
```

expensiveCall() always gets called,
even when we don’t use its result! 😱

Logging messages

```
func log(_ message: () -> String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    guard environment != .store else { return }

    print("[\(file):\(line)] \(function) - \(message())")
}
```

Logging messages

```
func log(_ message: () -> String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    guard environment != .store else { return }

    print("[\(file):\(line)] \(function) - \(message())")
}
```

Logging messages

```
func log(_ message: () -> String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    guard environment != .store else { return }

    print("[\(file):\(line)] \(function) - \(message())")
}

log({ "Result was: \(expensiveCall())" })
```

It gets the job done, but the extra syntax is really annoying 😢

Logging messages

```
func log(_ message: @autoclosure () -> String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    guard environment != .store else { return }

    print("[\(file):\(line)] \(function) - \(message())")
}
```

Logging messages

```
func log(_ message: @autoclosure () -> String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    guard environment != .store else { return }

    print("[\(file):\(line)] \(function) - \(message())")
}
```

Logging messages

```
func log(_ message: @autoclosure () -> String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    guard environment != .store else { return }

    print("[\(file):\(line)] \(function) - \(message())")
}
```

Logging messages

```
func log(_ message: @autoclosure () -> String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    guard environment != .store else { return }

    print("[\(file):\(line)] \(function) - \(message())")
}

log("Result was: \(expensiveCall())")
```

Logging messages

```
func log(_ message: @autoclosure () -> String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    guard environment != .store else { return }

    print("[\(file):\(line)] \(function) - \(message())")
}

log("Result was: \(expensiveCall())")
```

Logging messages

```
func log(_ message: @autoclosure () -> String,
         _ file: String = #file,
         _ line: Int = #line,
         _ function: String = #function) {
    guard environment != .store else { return }

    print("[\(file):\(line)] \(function) - \(message())")
}

log("Result was: \(expensiveCall())")
```

`@autoclosure` wraps an argument in a closure, so that it only gets evaluated if needed.