Swift with Hundreds of Engineers

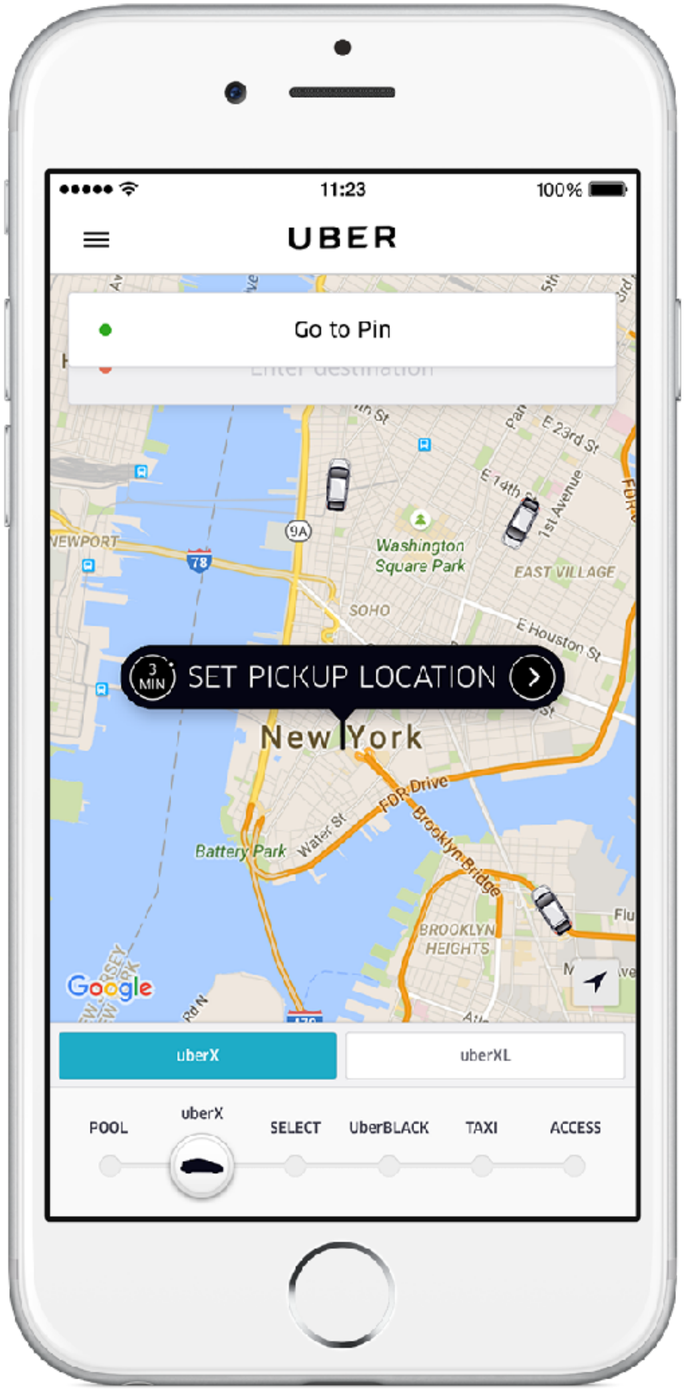
Tuomas Artman, Staff Engineer

May 13th, 2017

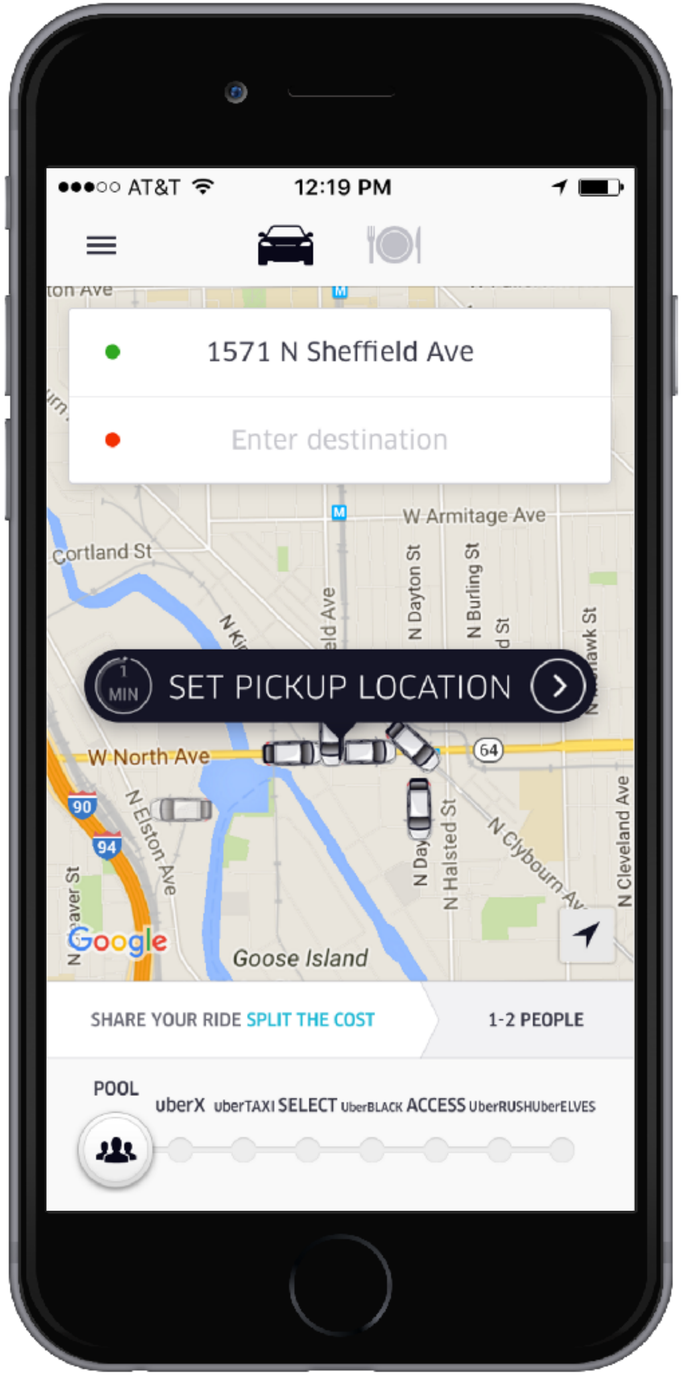
Swift with Hundreds of Engineers

Motivation, Architecture, Learnings

Uber’s mobile team 4 years ago


11:23
100%
UBER
Go to Pin
Enter destination
Washington Square Park
EAST VILLAGE
SOHO
E Houston St
3 MIN
SET PICKUP LOCATION
New York
FDR Drive
Water St
Battery Park
Brooklyn Bridge
BROOKLYN HEIGHTS
Google
uberX
uberXL
POOL
uberX
SELECT
UberBLACK
TAXI
ACCESS


11:23
100%
UBER
Go to Pin
Enter destination
Washington Square Park
EAST VILLAGE
SOHO
E Houston St
3 MIN
SET PICKUP LOCATION
New York
FDR Drive
Battery Park
Water St
Brooklyn Bridge
BROOKLYN HEIGHTS
Google
uberX
uberXL
POOL
uberX
SELECT
UberBLACK
TAXI
ACCESS


AT&T
12:19 PM
1571 N Sheffield Ave
Enter destination
W Armitage Ave
Cortland St
N Dayton St
N Burling St
1 MIN
SET PICKUP LOCATION
W North Ave
64
90
94
N Elston Ave
N Halsted St
N Cleveland Ave
Google
Goose Island
SHARE YOUR RIDE SPLIT THE COST
1-2 PEOPLE
POOL
uberX uberTAXI SELECT UberBLACK ACCESS UberRUSHUberELVES

“Let’s just change everything”

Rider App Rewrite

Architectural goals

99.99% reliability of core flows

Enable global roll-back of core flows to a guaranteed working state

Rider App Rewrite

Architectural goals

99.99% reliability of core flows

Enable global rollback of core flows to a guaranteed working state

Support Uber’s growth for years to come

Narrow and decouple functionality as much as possible

Rider App Rewrite

Architectural goals

99.99% reliability of core flows

Enable global rollback of core flows to a guaranteed working state

Support Uber’s growth for years to come

Narrow and decouple functionality as much as possible

Provide rails for both design and code

Guidelines for both architecture and design

Rider App Rewrite

Architectural goals

99.99% reliability of core flows

Enable global rollback of core flows to a guaranteed working state

Support Uber’s growth for years to come

Narrow and decouple functionality as much as possible

Provide rails for both design and code

Guidelines for both architecture and design

Monitoring is a first-class citizen

Automatic analytics, logging, debugging, and tracing

Rider App Rewrite

Architectural goals

99.99% reliability of core flows

Enable global rollback of core flows to a guaranteed working state

Support Uber’s growth for years to come

Narrow and decouple functionality as much as possible

Provide rails for both design and code

Guidelines for both architecture and design

Monitoring is a first-class citizen

Automatic analytics, logging, debugging, and tracing

De-risk experimentation

Application framework with plugin API

Rider App Rewrite

Architectural goals

99.99% reliability of core flows

Enable global rollback of core flows to a guaranteed working state

Support Uber's growth for years to come

Narrow and decouple functionality as much as possible

Provide rails for both design and code

Guidelines for both architecture and design

Monitoring is a first-class citizen

Automatic analytics, logging, debugging, and tracing

De-risk experimentation

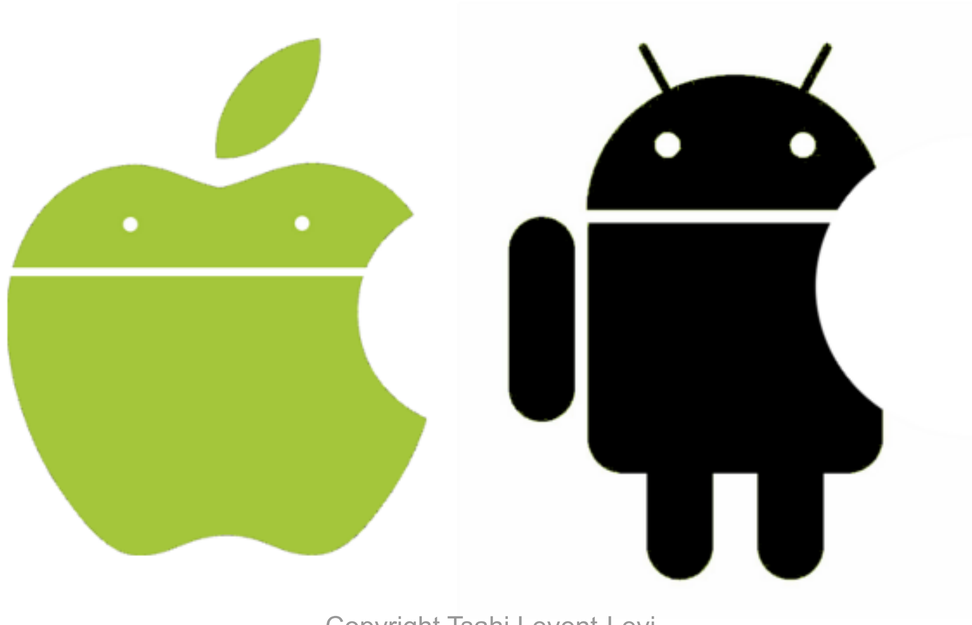
Application framework with plugin API

Make magic

Performance second to none, graceful degradation on low-end devices and networks

Multiplatform Architecture

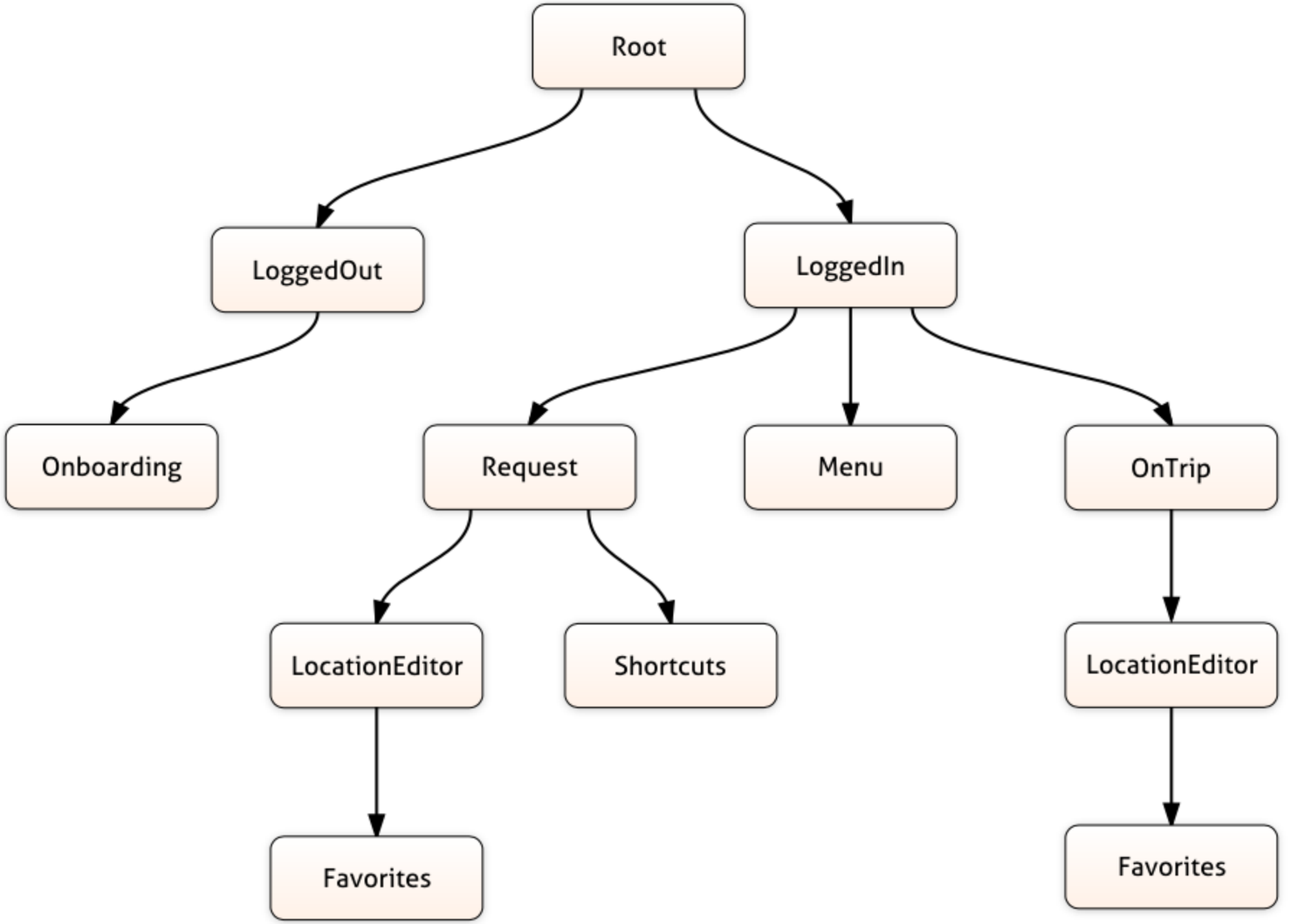
Double the effectiveness of your teams

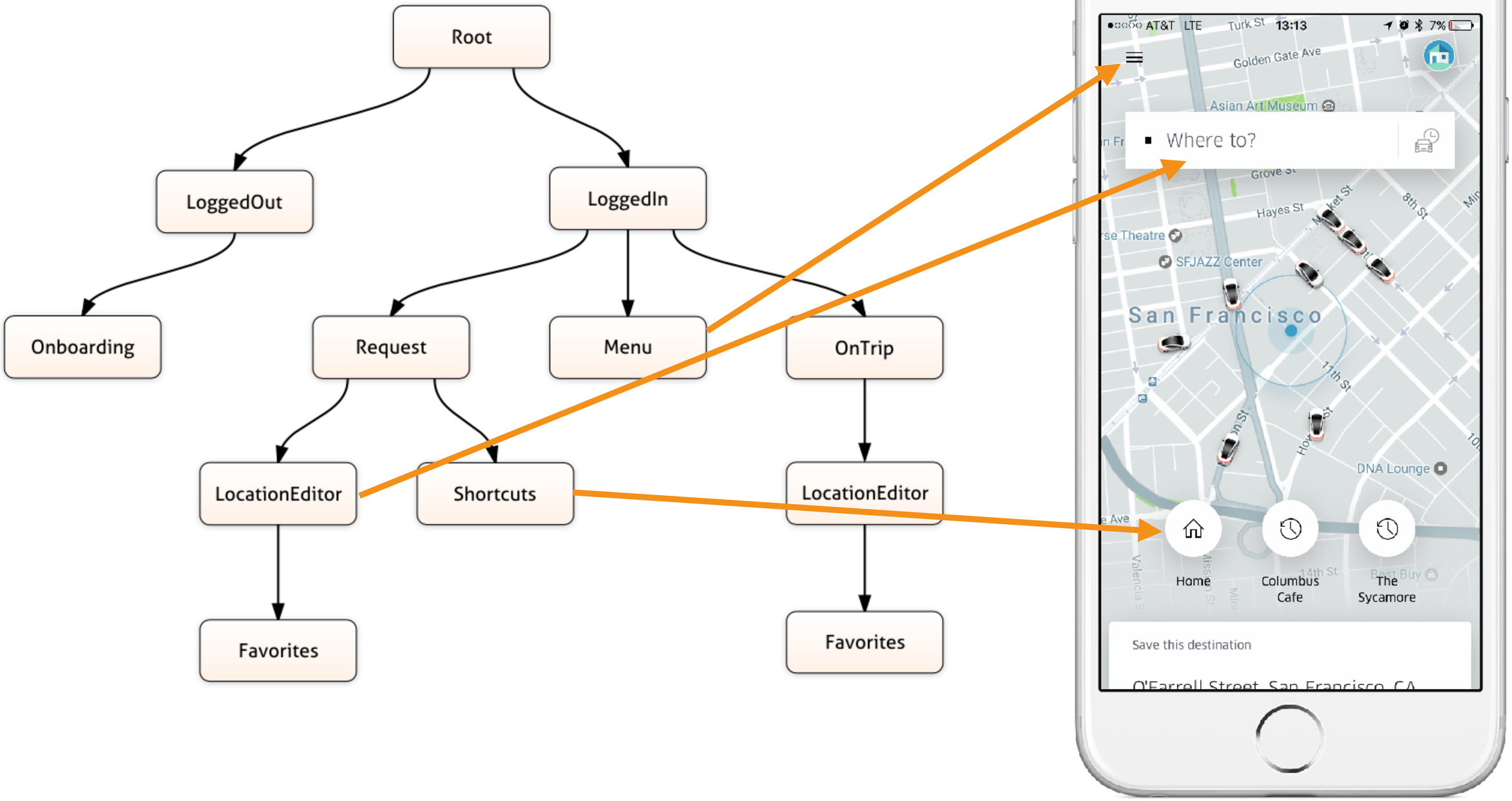


“RIBs”

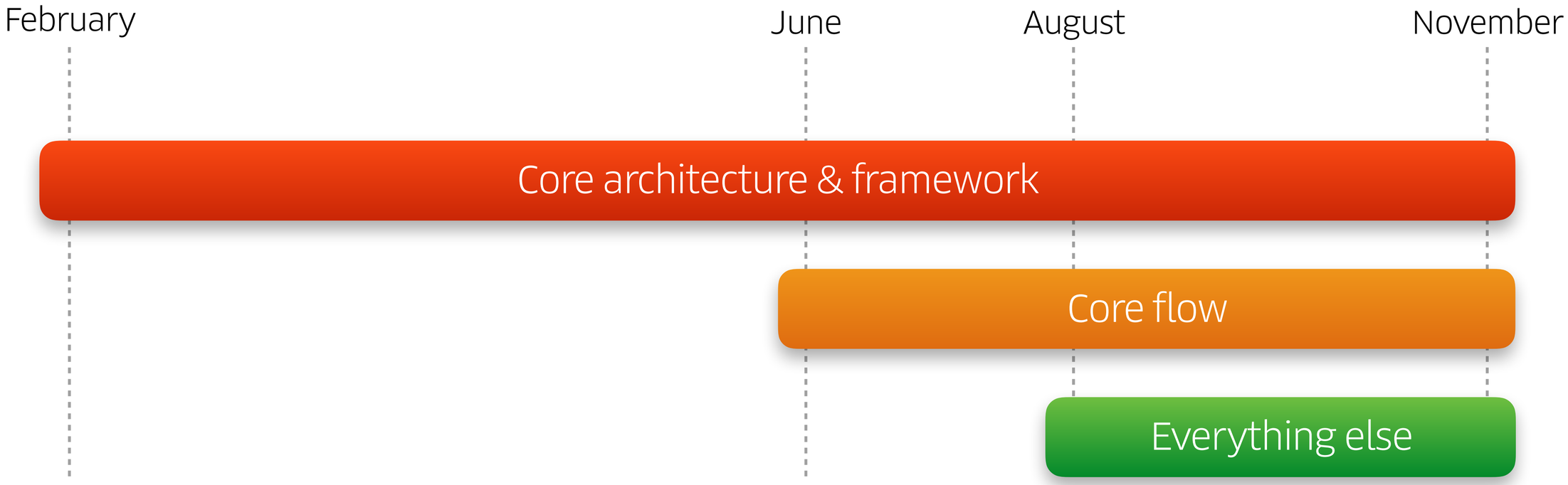
“RIBs”

Router Interaction Builder


Root
LoggedOut
LoggedIn
Onboarding
Request
Menu
OnTrip
LocationEditor
Shortcuts
LocationEditor
Favorites
Favorites


Root
LoggedOut
LoggedIn
Onboarding
Request
Menu
OnTrip
LocationEditor
Shortcuts
LocationEditor
Favorites
Favorites
AT&T LTE
13:13
7%
Where to?
Golden Gate Ave
Asian Art Museum
Hayes St
SFJAZZ Center
San Francisco
11th St
DNA Lounge
Home
Columbus Cafe
The Sycamore
Save this destination

Timeline


February
June
August
November
Core architecture & framework
Core flow
Everything else

Rider Application

A lot of files with a lot of lines of code

Over ten thousand Swift files

A million lines of Swift code

Lessons Learned

Swift – The good, the bad, and the ugly

Swift - the good

Defer

Guard

Functional programming

Tuples

Default parameters

Explicit overrides

It’s just a better

Generics

Type inference

Protocol extensions

Optionals

Less code

Enumerations

Explicit overrides

Readability

Swift - the good

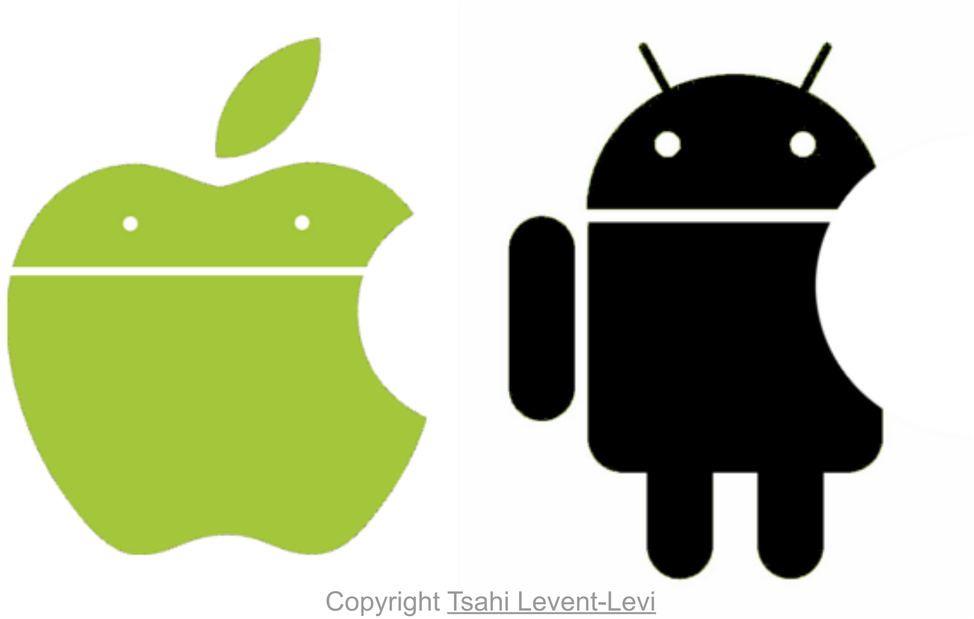
Reliability

Crash-free rate target:

99.99%

Swift - the good

Reliability

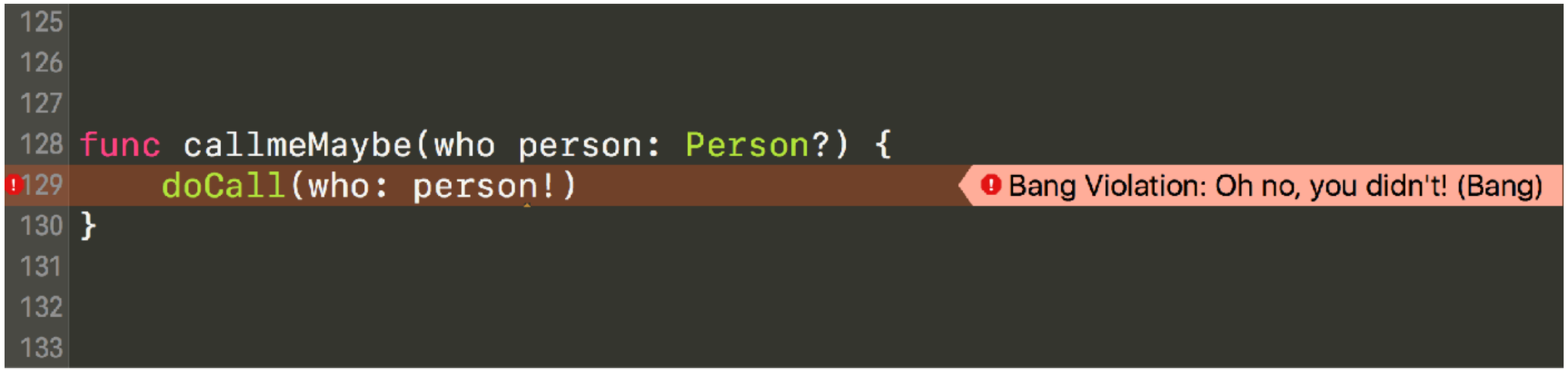


99.97% 99.90%

Swift beats Java by a factor of 3 in reliability*

Swift - the good

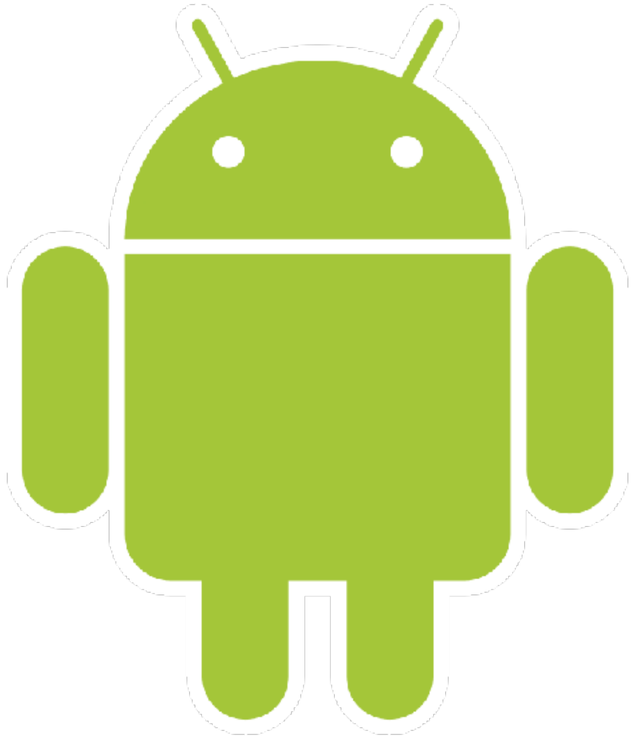
Reliability



Don’t unconditionally unwrap

Swift - the good

Android™ engineers



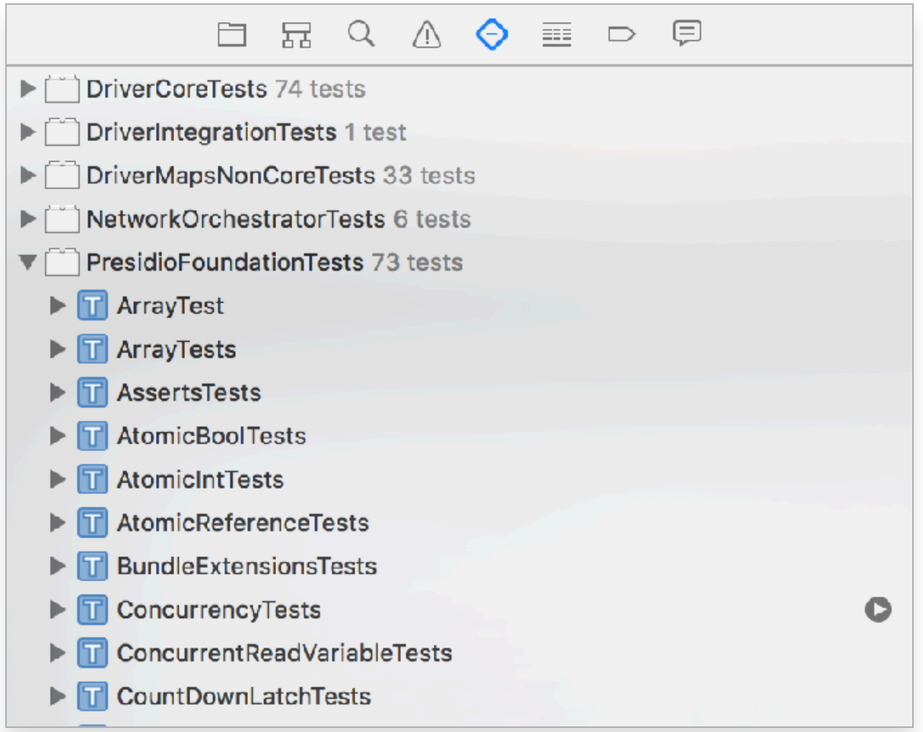
Android engineers more welcome!

Swift

The bad

Swift - the bad

Testing is hard



Testing

What can we do about it?

Testing is hard

```
/// @CreateMock
public protocol Storing {
    /// Fetches the data associated with `key`.
    ///
    /// - parameter key: the key whose data should be fetched.
    /// - parameter nameSpace: the nameSpace to retrieve the data from
    /// - returns: the data associated with the key, or nil if no data could be found
    public func dataForKey(key: String, nameSpace: String) -> Data?
    /// Stores `data`, associating it with `key`.
    ///
    /// - parameter key: the key to associate with `data`
    /// - parameter nameSpace: the nameSpace in which the association between `key` and `data` should be made
    /// - parameter data: the data to store
    /// - returns: the result of the storage operation.
    public func storeDataForKey(key: String, nameSpace: String, data: Data) -> Storage.StorageResult
}
```

What can we do about it?

Testing is hard

```
/// @CreateMock
public protocol Storing {
    /// Fetches the data associated with `key`.
    ///
    /// - parameter key: the key whose data should be fetched.
    /// - parameter nameSpace: the nameSpace to retrieve the data from
    /// - returns: the data associated with the key, or nil if no data could be found
    public func dataForKey(key: String, nameSpace: String) -> Data?
    /// Stores `data`, associating it with `key`.
    ///
    /// - parameter key: the key to associate with `data`
    /// - parameter nameSpace: the nameSpace in which the association between `key` and `data` should be made
    /// - parameter data: the data to store
    /// - returns: the result of the storage operation.
    public func storeDataForKey(key: String, nameSpace: String, data: Data) -> Storage.StorageResult
}
```

Mock generation:

```
artman@tuomas:~/Documents/ios$ script/generate-mocks
```

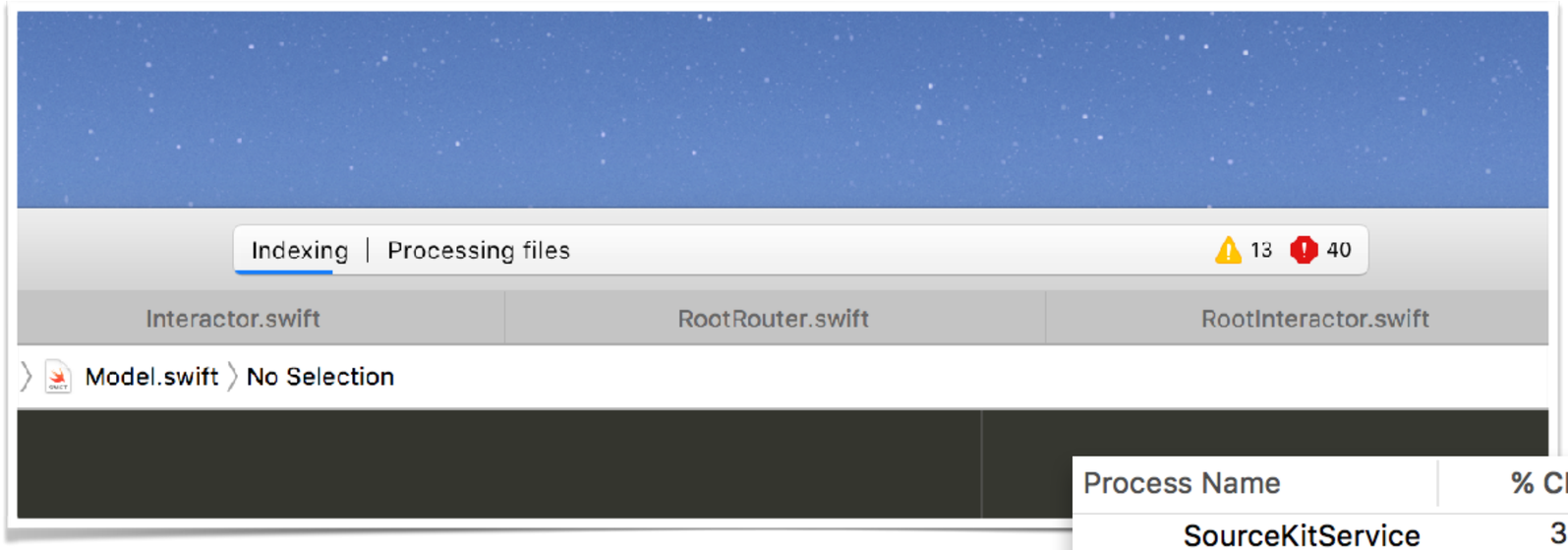
What can we do about it?

Testing is hard

```
/// A StoringMock class used for testing.
class StoringMock: Storing {

    // Function Handlers
    var dataForKeyHandler: ((key: String, nameSpace: String) -> (Data?))?
    var dataForKeyCallCount: Int = 0
    var storeDataForKeyHandler: ((key: String, nameSpace: String, data: Data) -> (StorageResult))?
    var storeDataForKeyCallCount: Int = 0

    init() {
    }

    func dataForKey(key: String, nameSpace: String) -> Data? {
        dataForKeyCallCount += 1
        if let dataForKeyHandler = dataForKeyHandler {
            return dataForKeyHandler(key: key, nameSpace: nameSpace)
        }
        // Default return type
        return nil
    }

    func storeDataForKey(key: String, nameSpace: String, data: Data) -> StorageResult {
        storeDataForKeyCallCount += 1
        if let storeDataForKeyHandler = storeDataForKeyHandler {
            return storeDataForKeyHandler(key: key, nameSpace: nameSpace, data: data)
        }
        // Default return type
        return StorageResult.Success
    }
}
```

Swift - the bad

Tooling issues

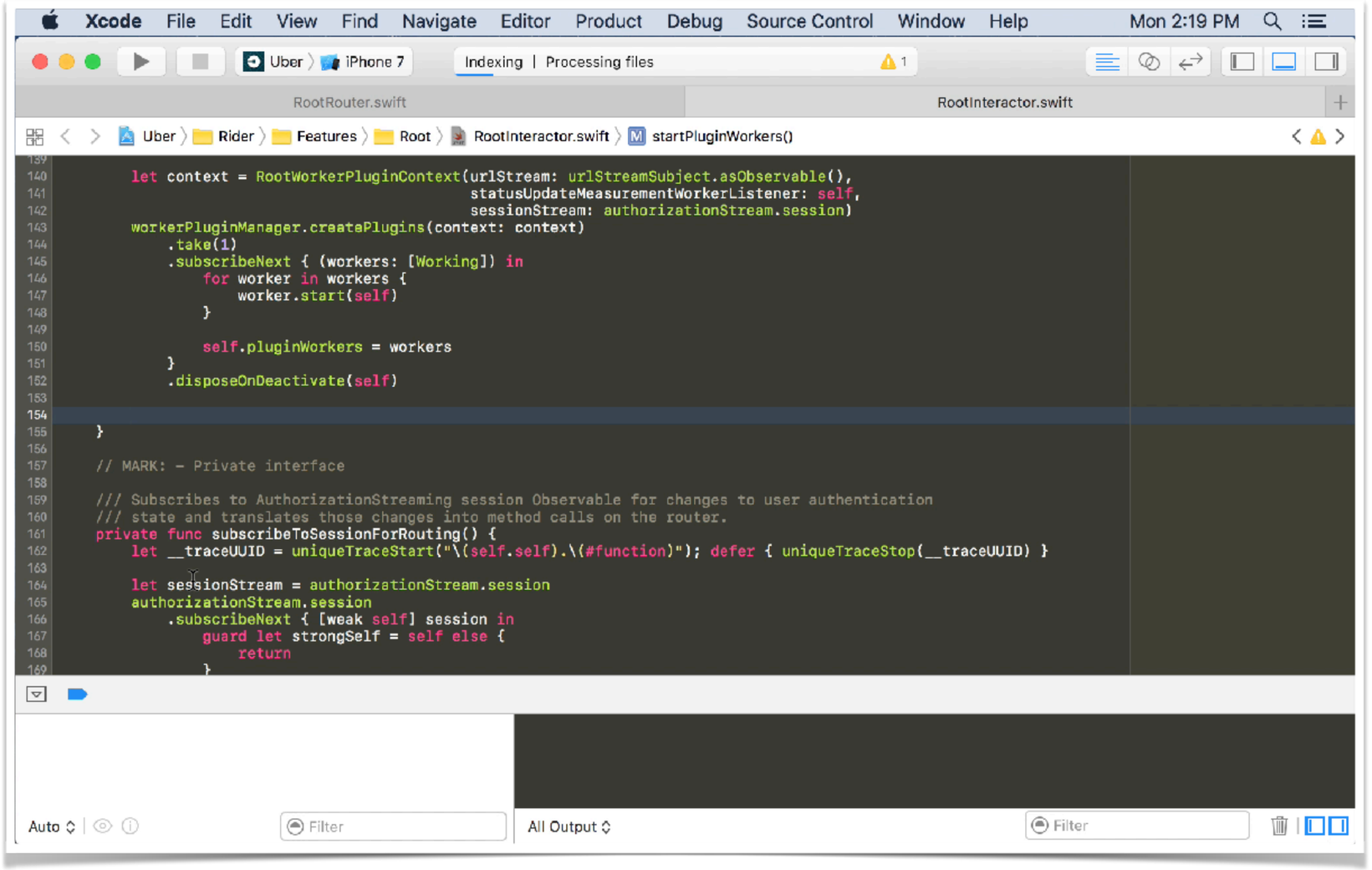


Infinity indexing

Process Name	% CPU	CPU Time
SourceKitService	328.3	4:25:51.16
com.apple.dt.So...	20.9	11:05.55
https://uber.hipc...	6.3	12:59.42

Swift - the bad

Tooling issues



What can we do about it?

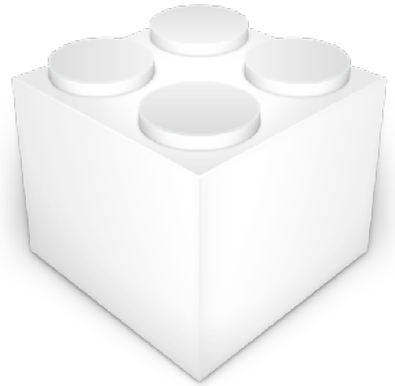
Tooling issues

What can we do about it?

Tooling issues

Upgrade!

Use alternatives



More frameworks

```
artman@tuomas:~/Documents/ios$ defaults write com.apple.dt.XCode IDEIndexDisable 1
```

Swift - the bad

Binary size

Any app's budget

Swift - the bad

Binary size

Structs

Struct are allocated on the stack and can increase binary size

Optionals

Are implemented as enums and add code that you might be unaware of

Generic specialization

Generics are awesome, but speed comes at a cost

Swift runtime libraries

4.5 MB for three architectures

What can we do about it?

Binary size

Wait for Swift 4

Apple is working on decreasing binary size of value types

Play around with optimization settings

Sometimes whole module optimization will yield smaller binary sizes, often larger

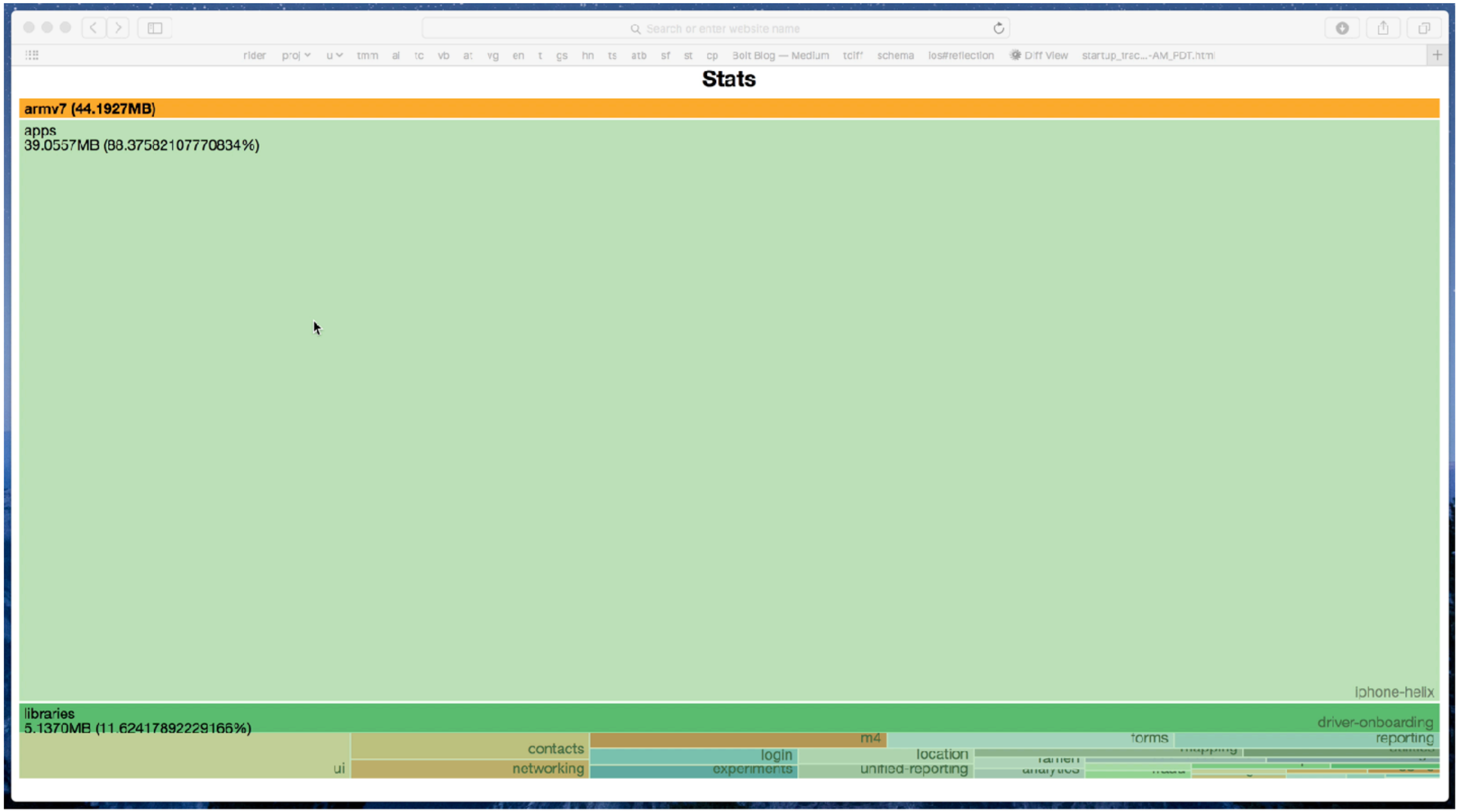
Know where you are spending binary size

We use link-maps to map symbols back to files

Then we combine all of them and generate an interactive tool

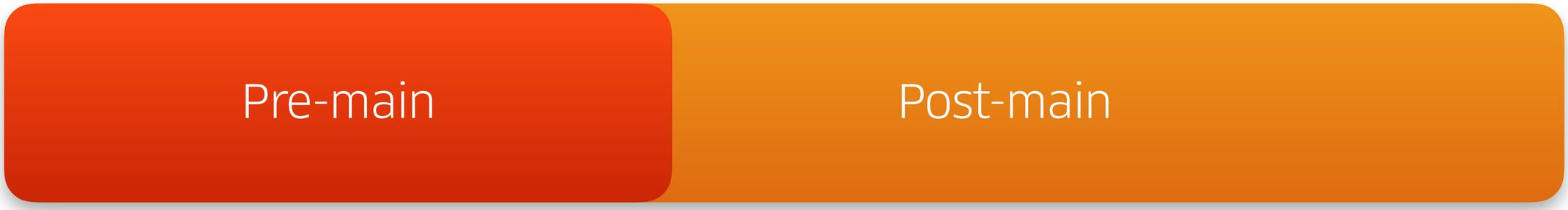
What can we do about it?

Binary size



Swift - the bad

Startup speed



What can we do about it?

Pre-main startup speed

Pre-main

The number of dynamic libraries linearly affects pre-main startup speed

- Re-link all of the symbols in all of our dynamic libraries into the application binary
- But you can’t link in the Swift runtime libraries (that’s 250ms on an iPhone 6s)

Test all the time, although its hard

- The number of dev/enterprise provisioning profiles on your phone greatly affects startup speed
- Tooling is needed to graph pre-main times

What can we do about it?

Post-main startup speed

Post-main

Reordering symbols in the app binary

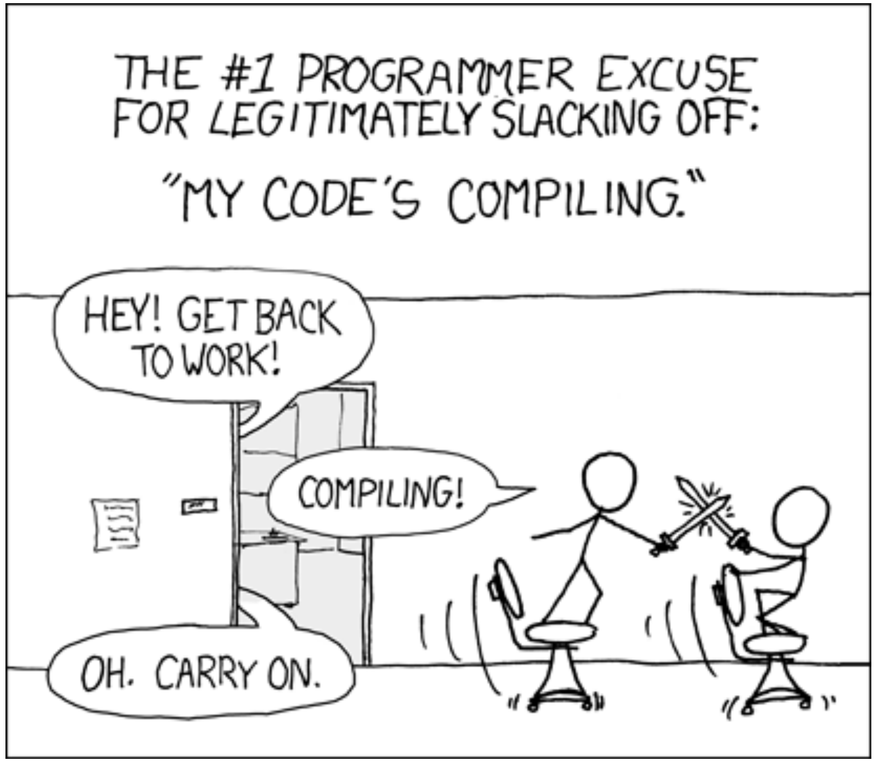
- Use DTrace to probe which symbols are accessed in your startup sequence and in what order
- Re-link your application with that order
- **20%** speedup on a 4s

Swift

The Ugly

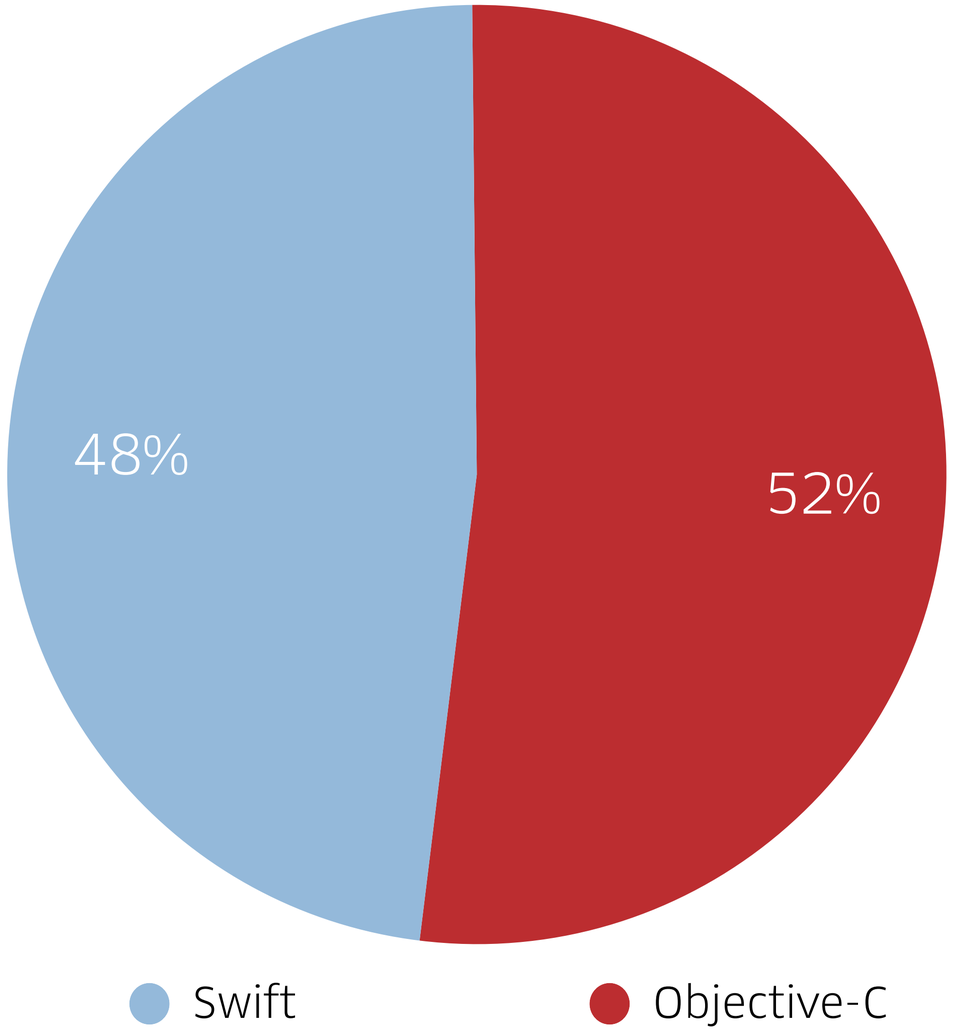
Swift - the ugly

Compile speeds

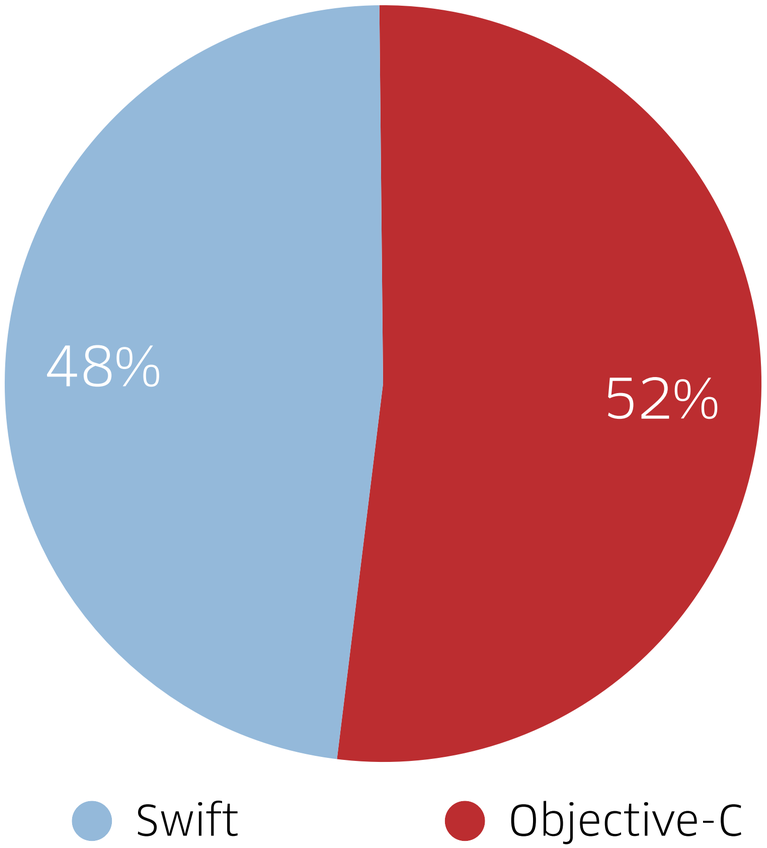


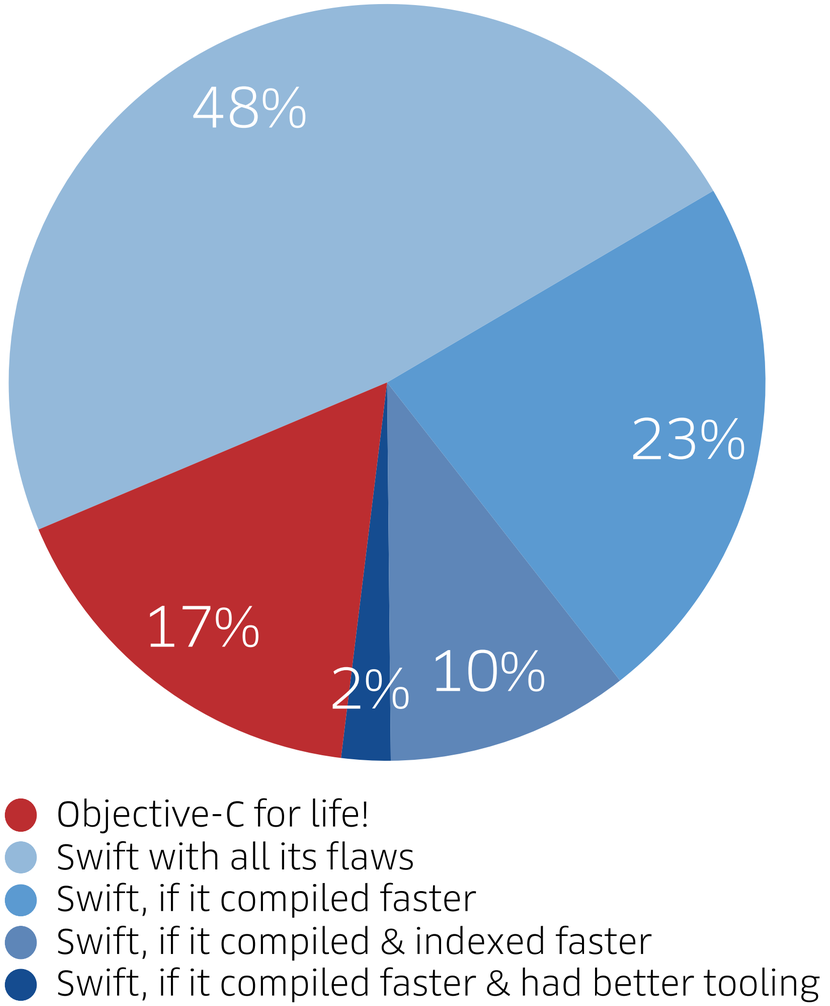
“Holistically considering all the positive and negative experiences you've had with writing code, which language do you think works better for iOS development at Uber going

June 2016



June 2016


48%
52%
Swift
Objective-C


48%
23%
17%
2%
10%
Objective-C for life!
Swift with all its flaws
Swift, if it compiled faster
Swift, if it compiled & indexed faster
Swift, if it compiled faster & had better tooling

What can we do about it?

Compile speeds

Contribute to Swift

- It’s open source

Add warnings on slow type inference

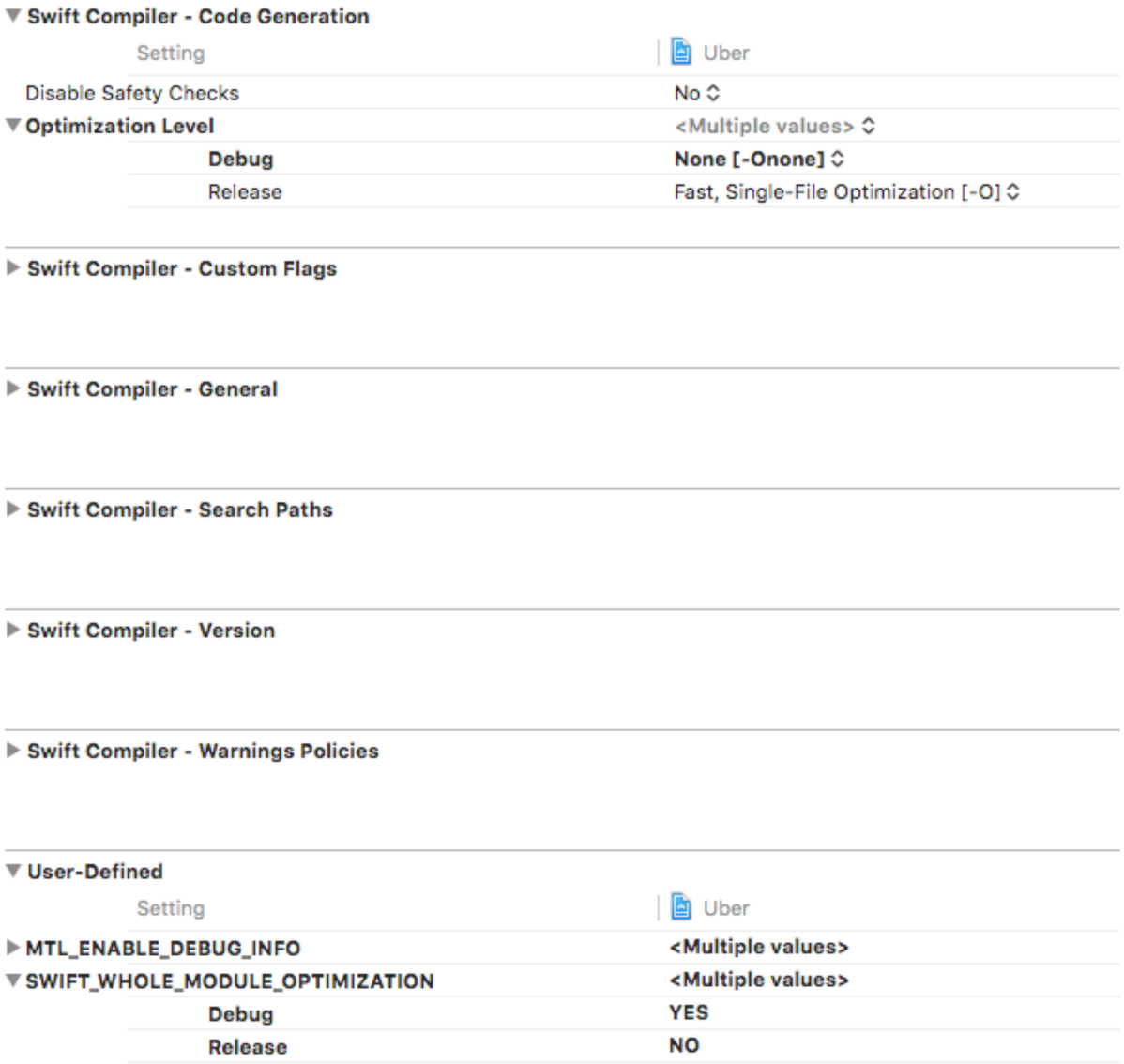
- Other Swift Flags: `-warn-long-function-bodies=100 -solver-memory-threshold 300000`

Combine files

- Merging 200 model files into 1 decreased compilation time from **1min 35sec** to **17sec**

What can we do about it?

Compile speeds



Buck

Superior dependency management

Reliable incremental builds

Remote build cache

https://buckbuild.com

Buck

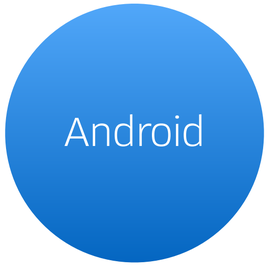
Superior dependency management

Reliable incremental builds

Remote build cache

Clean: **4x faster**

Incremental: **20x faster**



Clean: **6x faster**

Incremental: **30x faster**

Swift Support for Buck?

It’s (almost) here

Swift support for Xcode project file generation

- Implemented (https://github.com/facebook/buck/tree/uber-pr)

Swift support for Buck builds

- Implemented (https://github.com/facebook/buck/tree/uber-pr)

Swift support for Buck builds in the Xcode IDE

- Work not yet started…
- Generate projects based on what targets you want to work on
- Local builds can use the remote build cache

Results

How did Swift help us?

Rider app rewrite

Where does Swift help?

99.99% reliability of core flows

Enable global rollback of core flows to a guaranteed working state

Support Uber's growth for years to come

Narrow and decouple functionality as much as possible

Provide rails for both design and code

Guidelines for both architecture and design

Monitoring is a first-class citizen

Automatic analytics, logging, debugging, and tracing

De-risk experimentation

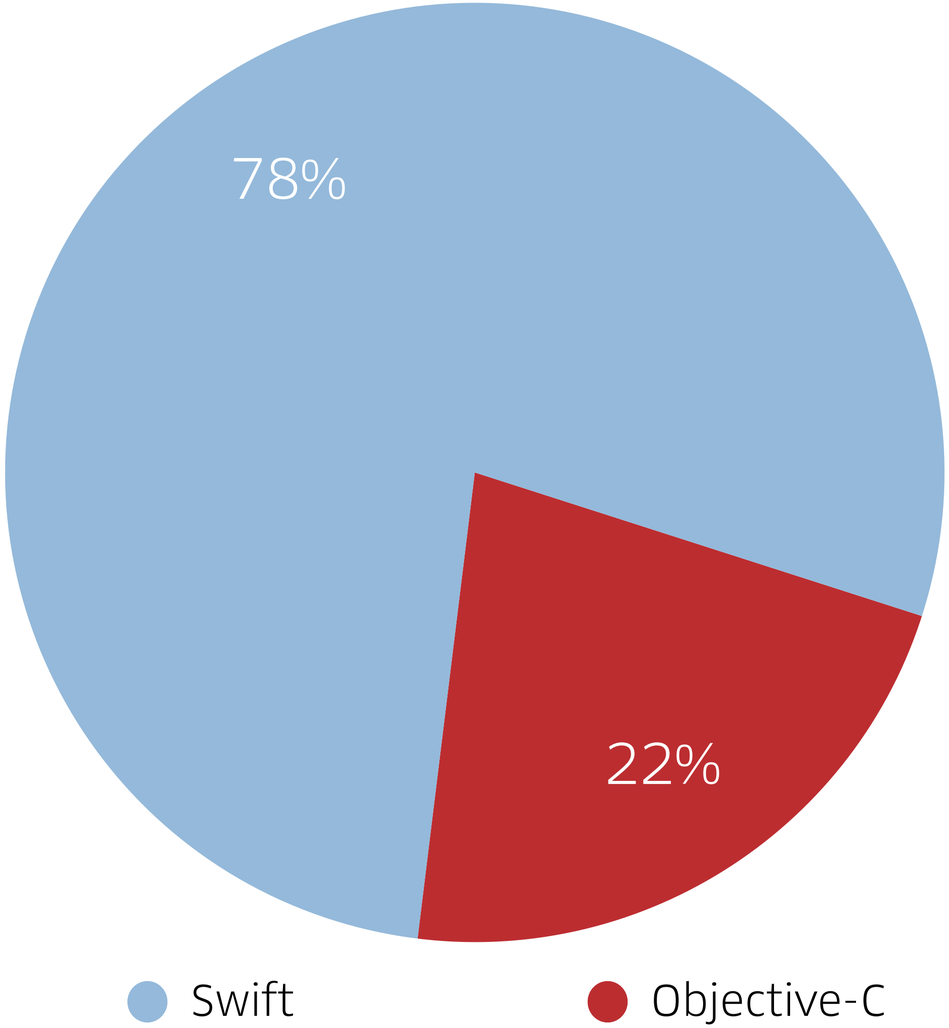
Application framework with plugin API

Make magic

Performance second to none, graceful degradation on low-end devices and networks

“Holistically considering all the positive and negative experiences you've had with writing code, which language do you think works better for iOS development at Uber going

February 2017



Takeaways

Takeaways

When growing your engineering team, make sure to:

- Keep an eye on compile times
- Monitor your binary sizes
- Figure out how to unit test
- Start using Buck

- When you start running into problems, your team should already be big enough to address these problems

uber.github.io

uber.github.io

eng.uber.com

Thank you!

Tuomas Artman

Mobile Platform, Uber
tuomas@uber.com
@artman

uber.github.io
eng.uber.com