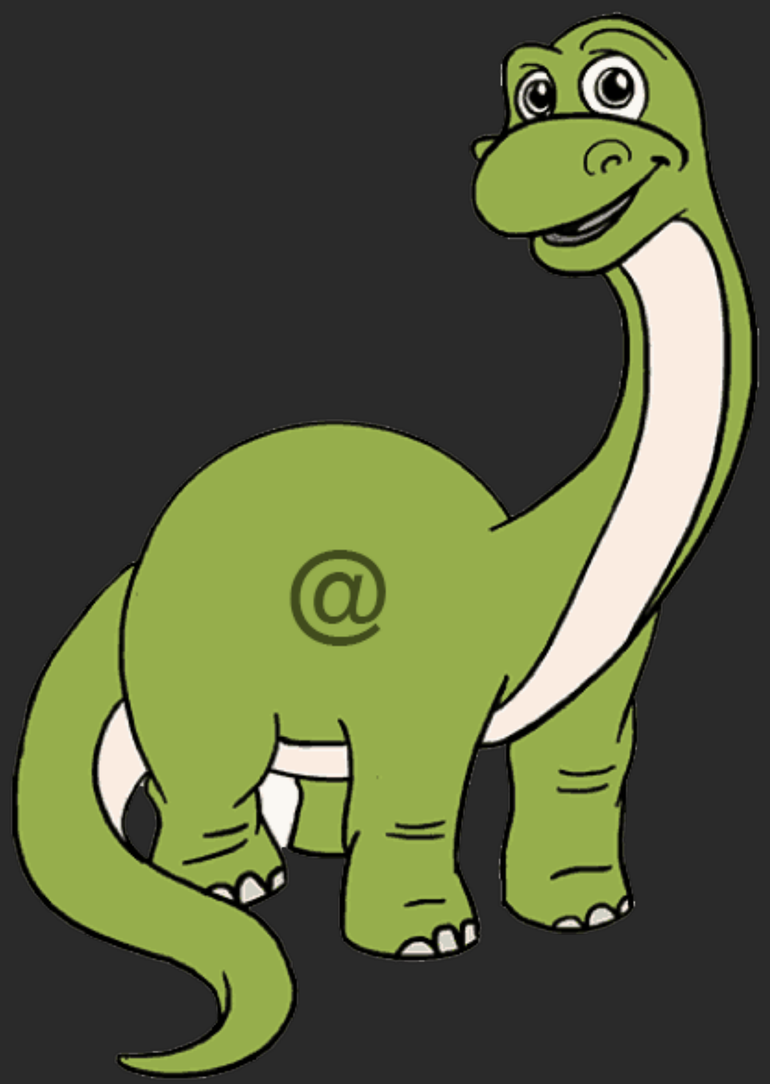
TAMING A MIXED CODEBASE

SWIFT & OBJECTIVE-C

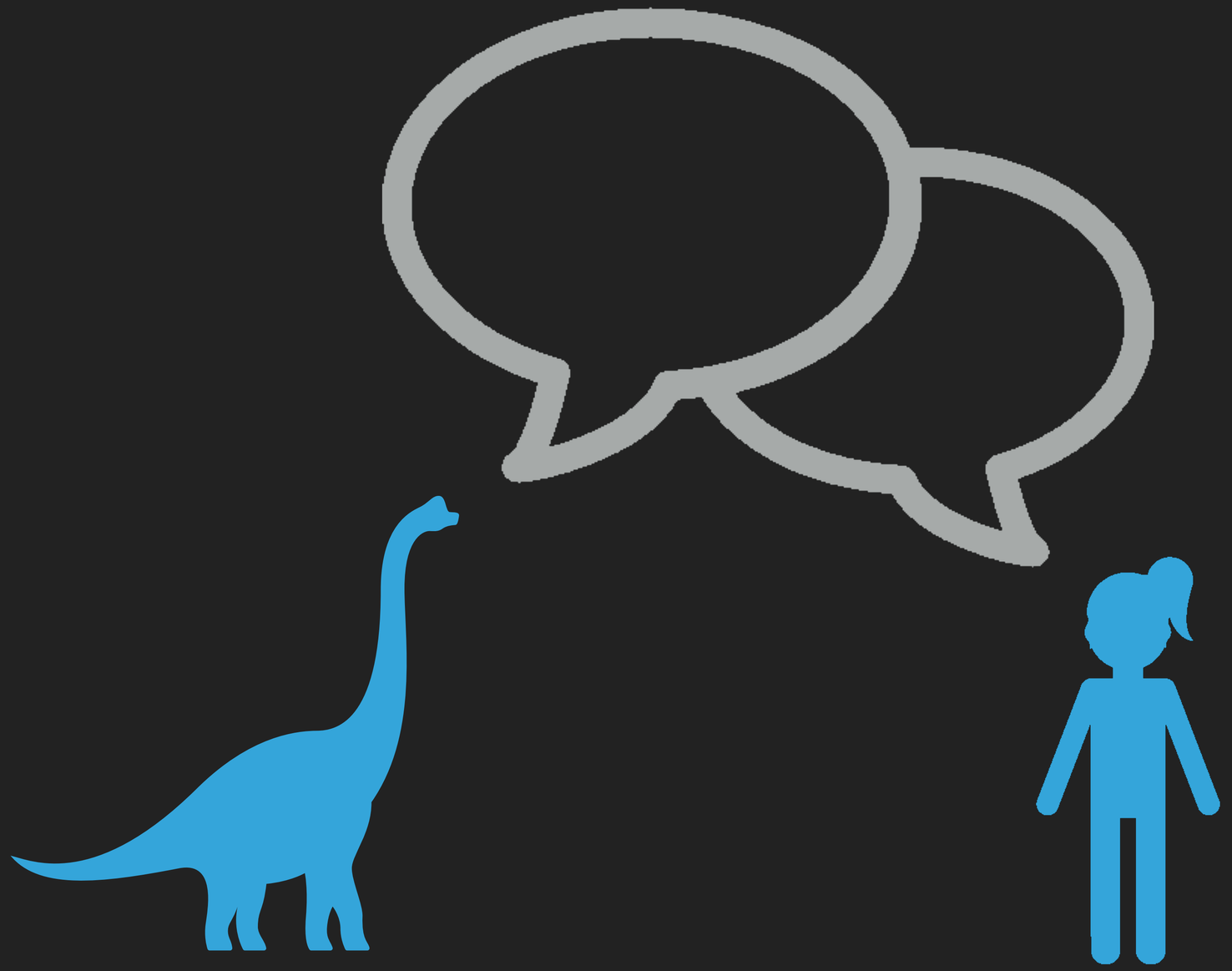
„FUNC-Y“

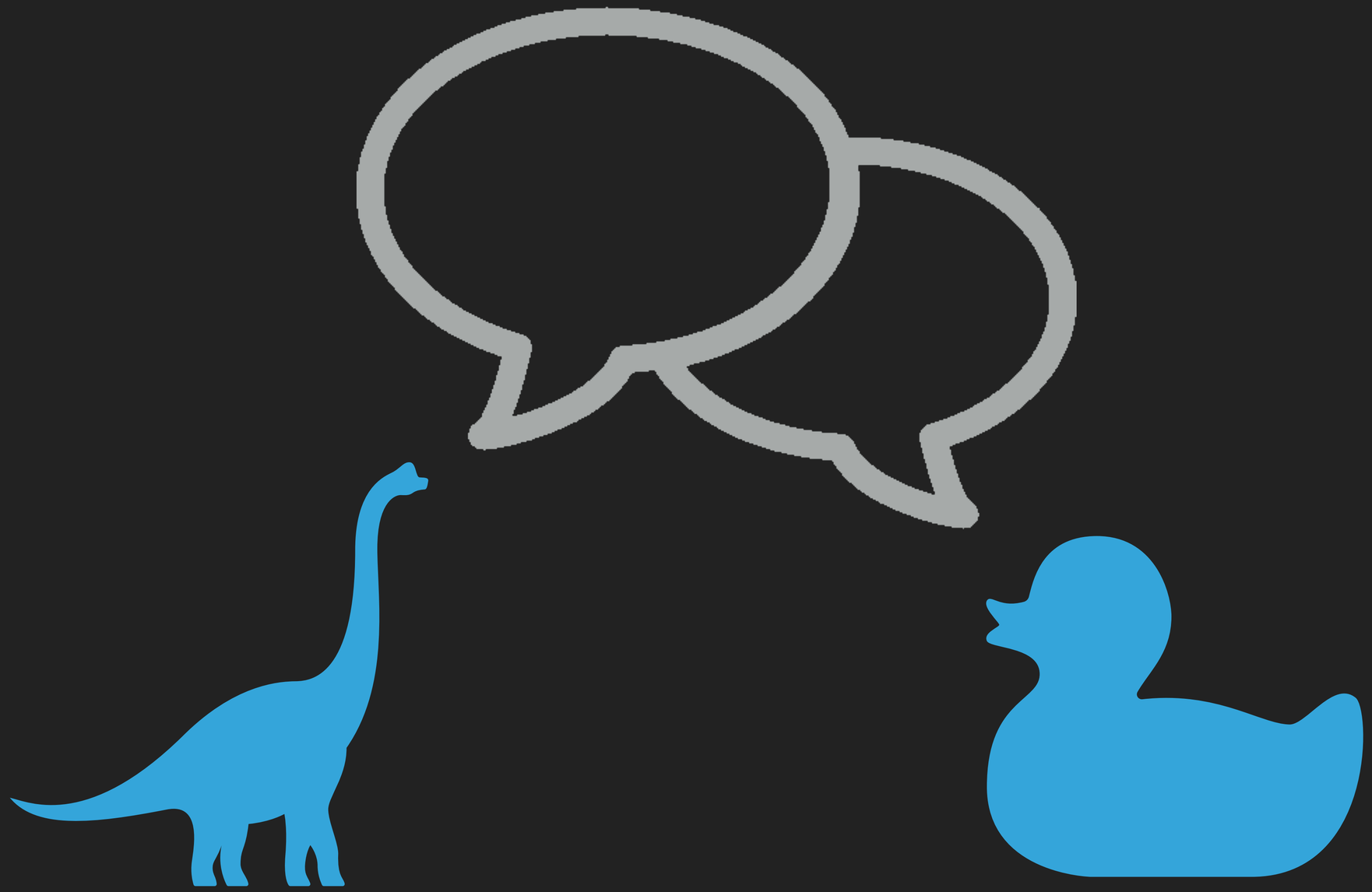
TAYLOR



„DYNAMIC“

DINO





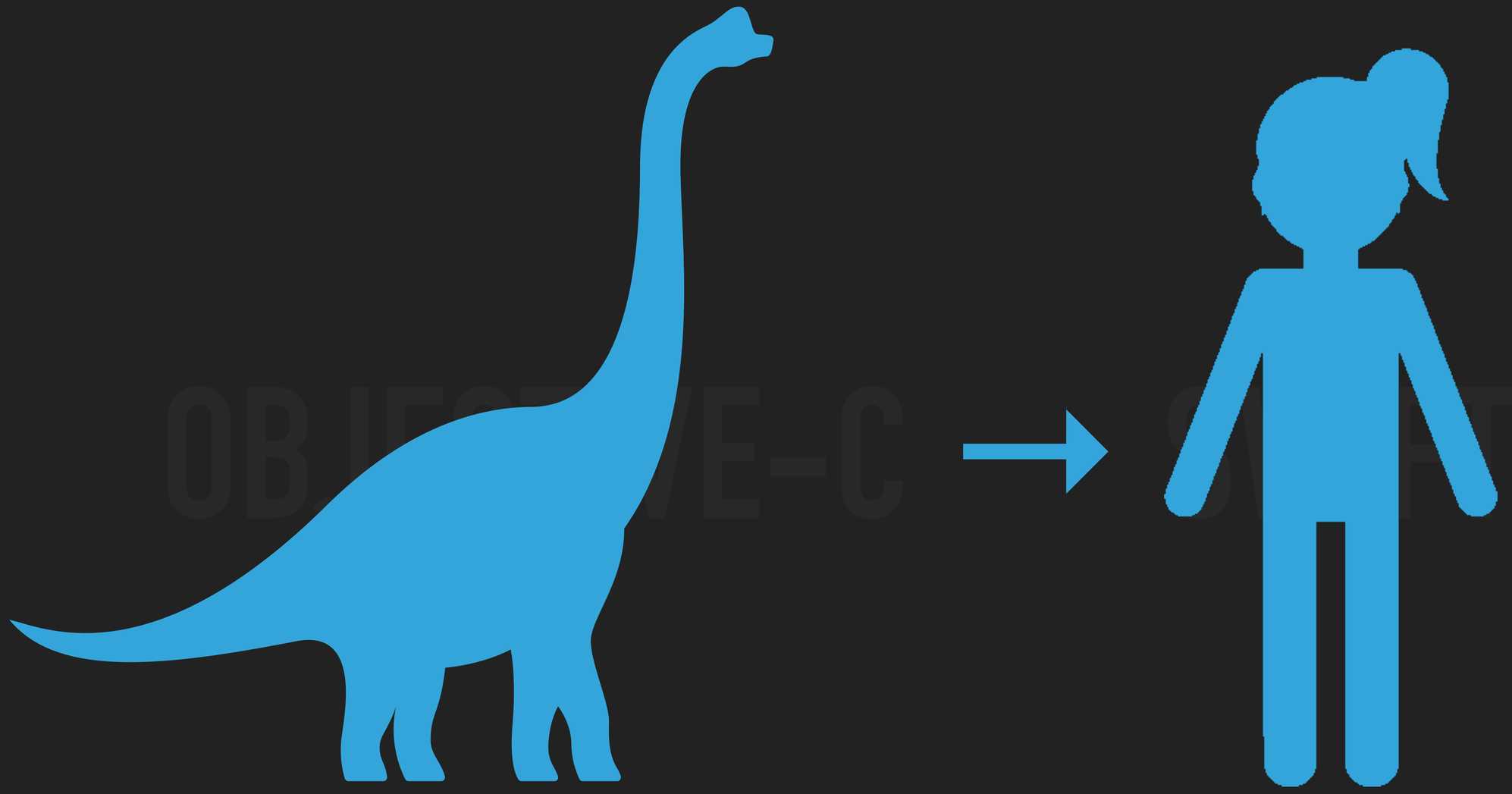
”

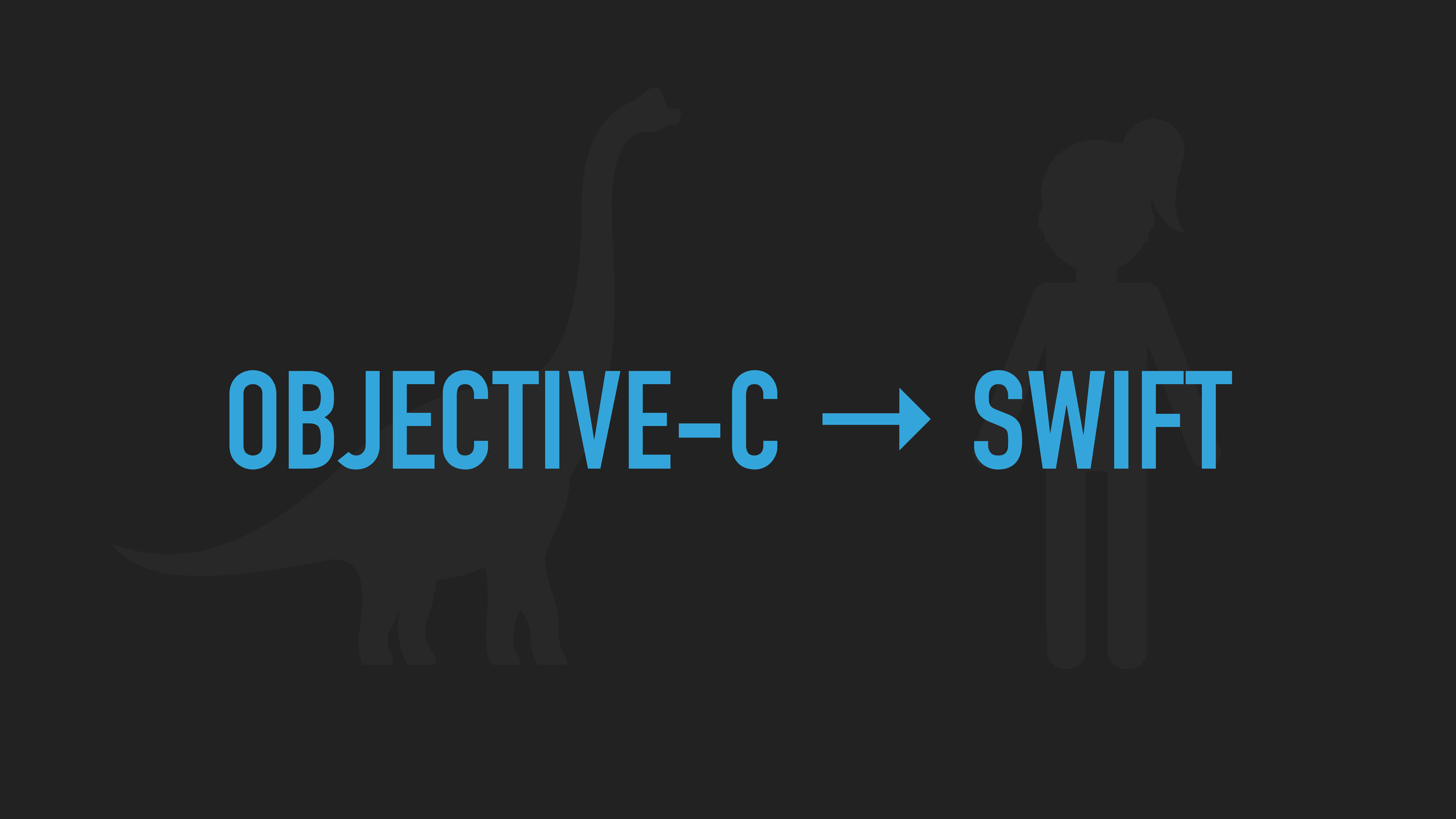
IN GENERAL,
I WOULD
SUMMARIZE THE
EXPERIENCE WITH
THIS PICTURE

@meghafon

TAMING A MIXED CODEBASE

SWIFT & OBJECTIVE-C




OBJECTIVE-C → SWIFT

INTENTION & CLARITY

CLASS PROPERTIES

NS_ENUM

LIGHTWEIGHT GENERICS

NS_OPTIONS

INTENTION & CLARITY

INSTANCETYPE

PROPERTIES

NS_DESIGNATED_INITIALIZER

NULLABILITY

INTEROPERABILITY

NS_SWIFT_NAME

NS_TYPED_ENUM

NS_SWIFT_UNAVAILABLE

INTEROPERABILITY

NS_TYPED_EXTENSIBLE_ENUM

NS_SWIFT_NOTHROW

NS_REFINED_FOR_SWIFT

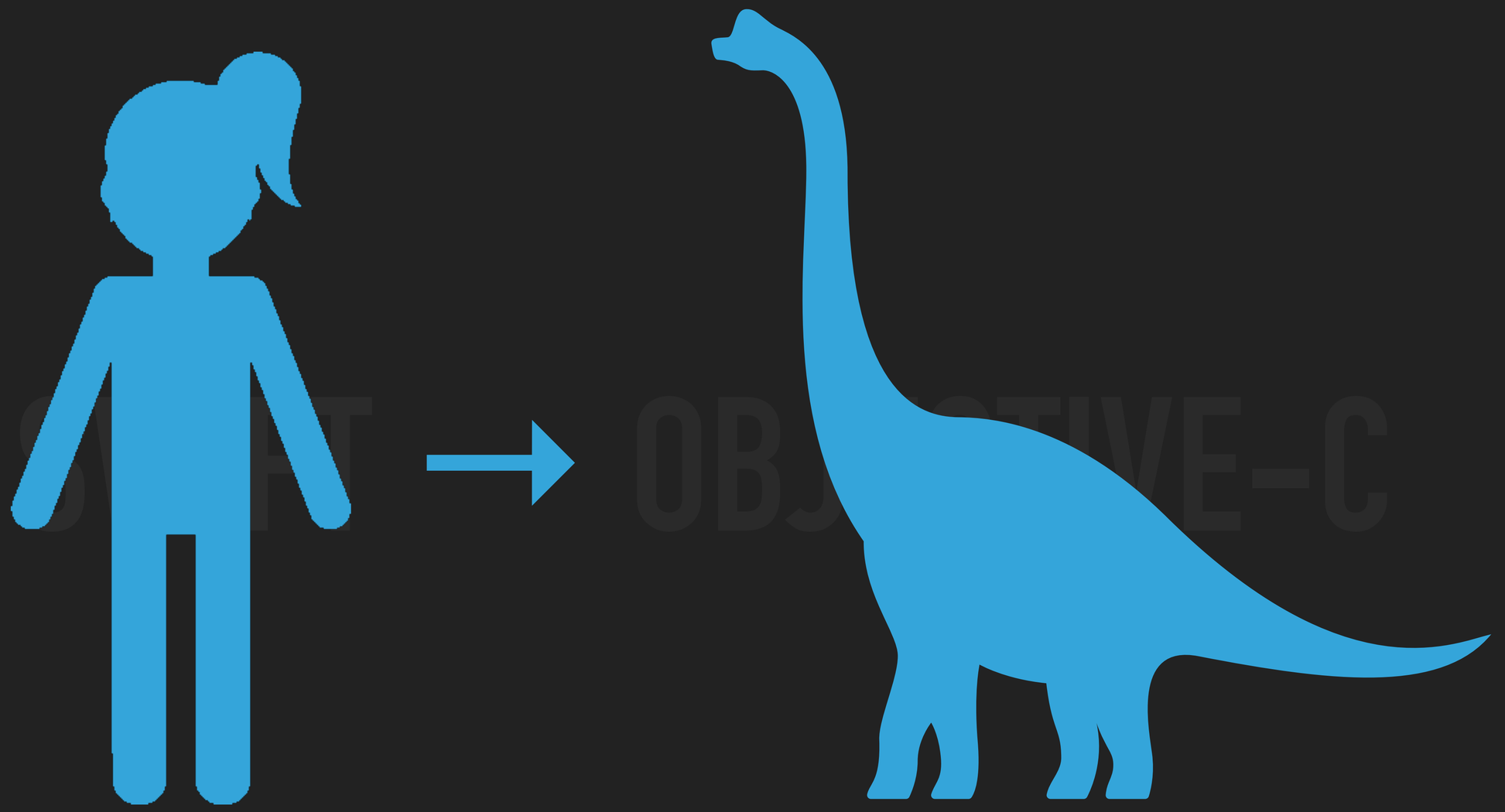
CONVENIENCE

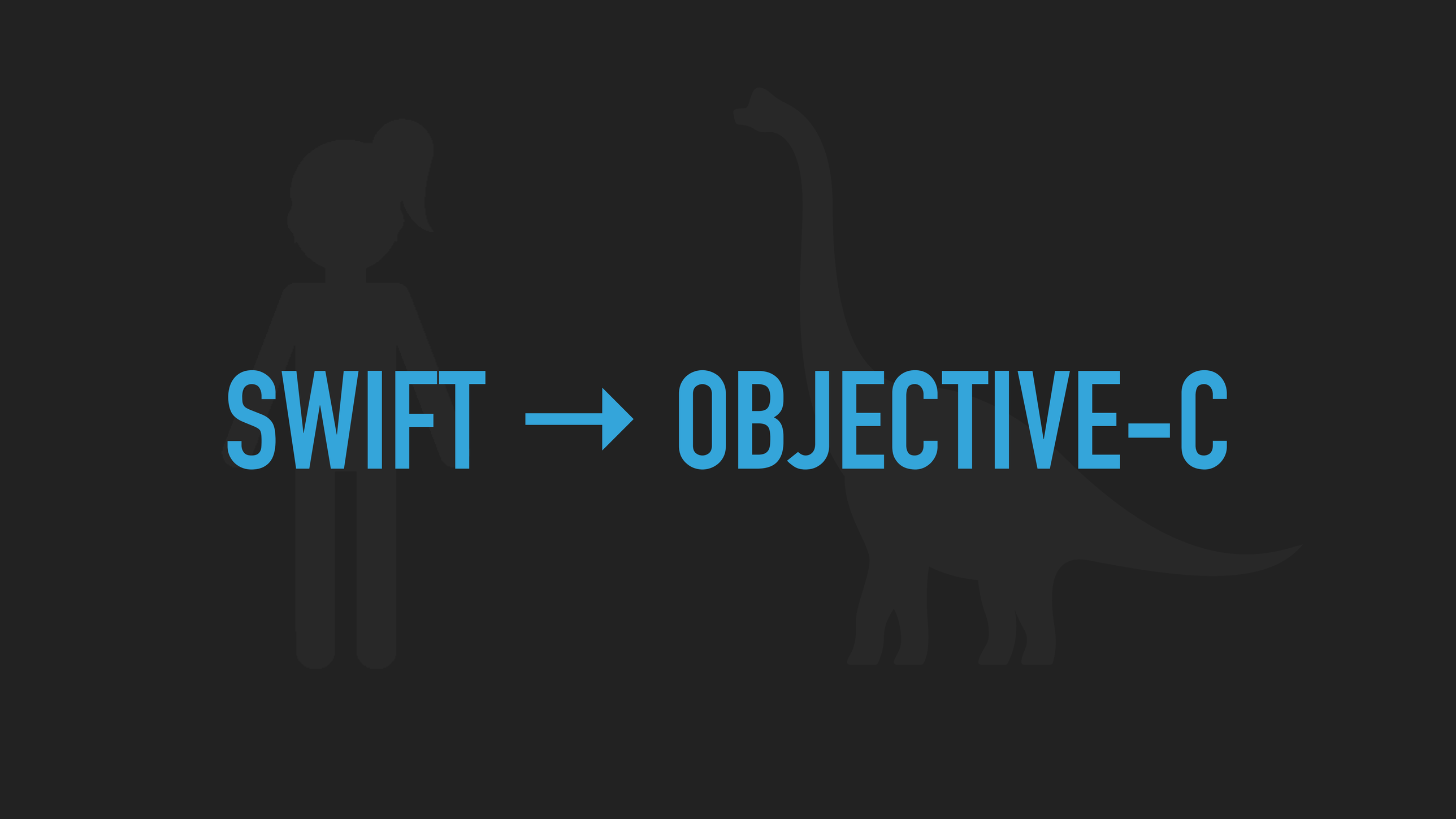
```
#define var __auto_type
#define let const __auto_type
```

https://pspdfkit.com/blog/2018/first-class-swift-api-for-objective-c-frameworks/

Marcin Krzyżanowski

DEMO


SWIFT
OBJECTIVE-C


SWIFT → OBJECTIVE-C

@objc

@objc(newName:)

„but what about my tuples, enums, structs?“

NO.

```
@objc final class NodeDraggingState: NSObject {
    enum Mode {
        case dragging(nodeIDs: Set<ObjectID>)
        case nodeCreation(viewModel: NodeViewModel)
    }

    init(mode: Mode, location: CGPoint) { … }

    @objc @available(swift, obsoleted: 1.0)
    convenience init(nodeIDs: Set<ObjectID>, location: CGPoint) {
        self.init(mode: .dragging(nodeIDs: nodeIDs),
                  location: location)
    }

    @objc @available(swift, obsoleted: 1.0)
    convenience init(viewModel: NodeViewModel, location: CGPoint) {
        self.init(mode: .nodeCreation(viewModel: viewModel),
                  location: location)
    }
}
```

```
@objc final class NodeDraggingState: NSObject {
    enum Mode {
        case dragging(nodeIDs: Set<ObjectID>)
        case nodeCreation(viewModel: NodeViewModel)
    }

    init(mode: Mode, location: CGPoint) { … }

    @objc @available(swift, obsoleted: 1.0)
    convenience init(nodeIDs: Set<ObjectID>, location: CGPoint) {
        self.init(mode: .dragging(nodeIDs: nodeIDs),
                  location: location)
    }

    @objc @available(swift, obsoleted: 1.0)
    convenience init(viewModel: NodeViewModel, location: CGPoint) {
        self.init(mode: .nodeCreation(viewModel: viewModel),
                  location: location)
    }
}
```

```
@objc final class NodeDraggingState: NSObject {
    enum Mode {
        case dragging(nodeIDs: Set<ObjectID>)
        case nodeCreation(viewModel: NodeViewModel)
    }

    init(mode: Mode, location: CGPoint) { … }

    @objc @available(swift, obsoleted: 1.0)
    convenience init(nodeIDs: Set<ObjectID>, location: CGPoint) {
        self.init(mode: .dragging(nodeIDs: nodeIDs),
                  location: location)
    }

    @objc @available(swift, obsoleted: 1.0)
    convenience init(viewModel: NodeViewModel, location: CGPoint) {
        self.init(mode: .nodeCreation(viewModel: viewModel),
                  location: location)
    }
}
```

```
@objc final class NodeDraggingState: NSObject {
    enum Mode {
        case dragging(nodeIDs: Set<ObjectID>)
        case nodeCreation(viewModel: NodeViewModel)
    }

    init(mode: Mode, location: CGPoint) { … }

    @objc @available(swift, obsoleted: 1.0)
    convenience init(nodeIDs: Set<ObjectID>, location: CGPoint) {
        self.init(mode: .dragging(nodeIDs: nodeIDs),
                  location: location)
    }

    @objc @available(swift, obsoleted: 1.0)
    convenience init(viewModel: NodeViewModel, location: CGPoint) {
        self.init(mode: .nodeCreation(viewModel: viewModel),
                  location: location)
    }
}
```

DRAGON ZONE

DRAGON ZONE

_ObjectiveCBridgeable

DRAGON ZONE

Non-optional Parameters

```
func assertNotNil<T>(_ t: T) -> Bool {
    assert(t as T? != nil,
           “var unexpectedly 'nil',
           fooled by ObjC?")

    return t as T? != nil
}
```

```
func assertNotNil<T>(_ t: T) -> Bool {
    assert(t as T? != nil,
           “var unexpectedly 'nil',
           fooled by ObjC?")

    return t as T? != nil
}
```

DRAGON ZONE

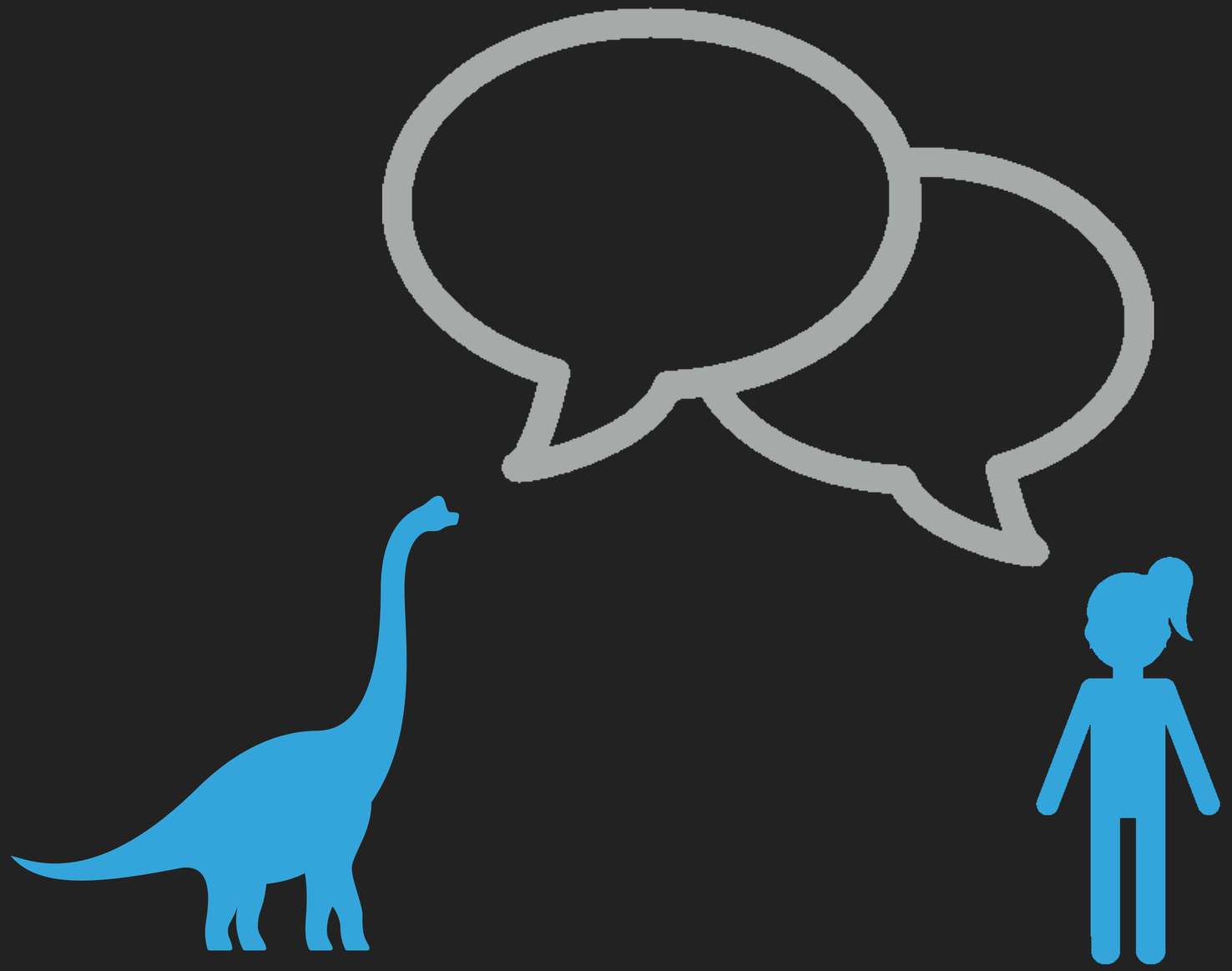
Bridging Casts from *AnyHashable*

```
class CustomEntry: NSObject { }

while true {
    let x = CustomEntry()
    let y: AnyHashable = x
    print(y as! CustomEntry)
}
```

```
class CustomEntry: NSObject { }

while true {
    let x = CustomEntry()
    let y: AnyHashable = x
    print(y as! CustomEntry)
}
```



@myell0w(.com)

THANK YOU