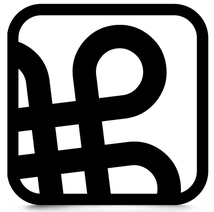


Macoun

Swifter Code in Objective-C(++)

Hintergrund

Hintergrund

- Michael Ochs
- iOS Entwickler seit iOS 3
- PSPDFKit GmbH

Hintergrund

- Wir entwickeln ein Framework für über 1000 Kunden
  - Integriert in Apps von Dropbox, Evernote, Atlassian, …
- Grosser Quelltext (> 600.000 loc)
- Grosse öffentliche Schnittstelle (> 300 classes, > 70 protocols)
- Distribuiert in Binärform

Die Probleme mit Swift

Die Probleme mit Swift

- Keine Quelltext Kompatibilität
- Keine Binär Kompatibilität
- Immer noch Fehler in der Syntax

Die Probleme mit Swift

- Quelltext muss weiter funktionieren ohne Änderungen
- Schnittstelle muss so stabil wie möglich bleiben
- Distribution muss schnell und einfach möglich sein
- Unterstützung der letzten zwei iOS Versionen

Die Probleme mit Swift

Quelltext Kompatibilität

- Breaking changes im Quelltext sind sehr kostspielig
- Die meisten Stellen in jedem Quelltext werden selten geändert
- Schreiben von Code erzeugt immer Fehler
  - Code zu ändern der vorher funktionierte erzeugt Fehler ohne erkennbaren Vorteil

Die Probleme mit Swift

- Keine ABI Kompatibilität
- Keine std library Kompatibilität
- Nicht nutzbar mit C++

Objective-C swifter machen

Objective-C swifter machen

- Boiler plate code vermeiden
- Eigene Helfer-Kategorien schreiben
  - An Swift orientieren
- Helfer in Frameworks modularisieren
  - Sehr einfach über private CocoaPods zu lösen

Objective-C swifter machen

```
@interface NSArray<ObjectType> (Helper)

- (NSArray<ObjectType> *)pspdf_filtered:
                              (NS_NOESCAPE BOOL(^)(__kindof ObjectType obj))block;

- (NSArray *)pspdf_mapped:
                       (NS_NOESCAPE id _Nullable (^)(__kindof ObjectType obj))block;

- (NSArray *)pspdf_flattened;

- (NSArray *)pspdf_mutatedArrayUsingBlock:
          (NS_NOESCAPE void (^)(NSMutableArray<__kindof ObjectType> *array))block;

@end
```

Objective-C swifter machen

```
@interface NSArray<ObjectType> (Helper)

- (NSArray<ObjectType> *)pspdf_filtered:
                              (NS_NOESCAPE BOOL(^)(__kindof ObjectType obj))block;

- (NSArray *)pspdf_mapped:
                       (NS_NOESCAPE id _Nullable (^)(__kindof ObjectType obj))block;

- (NSArray *)pspdf_flattened;

- (NSArray *)pspdf_mutatedArrayUsingBlock:
          (NS_NOESCAPE void (^)(NSMutableArray<__kindof ObjectType> *array))block;

@end
```

Objective-C swifter machen

```
@interface NSArray<ObjectType> (Helper)

- (NSArray<ObjectType> *)pspdf_filtered:
                              (NS_NOESCAPE BOOL(^)(__kindof ObjectType obj))block;

- (NSArray *)pspdf_mapped:
                       (NS_NOESCAPE id _Nullable (^)(__kindof ObjectType obj))block;

- (NSArray *)pspdf_flattened;

- (NSArray *)pspdf_mutatedArrayUsingBlock:
          (NS_NOESCAPE void (^)(NSMutableArray<__kindof ObjectType> *array))block;

@end
```

Objective-C swifter machen

```
@interface NSArray<ObjectType> (Helper)

- (NSArray<ObjectType> *)pspdf_filtered:
                              (NS_NOESCAPE BOOL(^)(__kindof ObjectType obj))block;

- (NSArray *)pspdf_mapped:
                       (NS_NOESCAPE id _Nullable (^)(__kindof ObjectType obj))block;

- (NSArray *)pspdf_flattened;

- (NSArray *)pspdf_mutatedArrayUsingBlock:
          (NS_NOESCAPE void (^)(NSMutableArray<__kindof ObjectType> *array))block;

@end
```

Objective-C swifter machen

```
@interface NSArray<ObjectType> (Helper)

- (NSArray<ObjectType> *)pspdf_filtered:
                              (NS_NOESCAPE BOOL(^)(__kindof ObjectType obj))block;

- (NSArray *)pspdf_mapped:
                       (NS_NOESCAPE id _Nullable (^)(__kindof ObjectType obj))block;

- (NSArray *)pspdf_flattened;

- (NSArray *)pspdf_mutatedArrayUsingBlock:
          (NS_NOESCAPE void (^)(NSMutableArray<__kindof ObjectType> *array))block;

@end
```

Demo

Objective-C++

Objective-C++

- Die ganze Macht von C++ in Objective-C Dateien
- Aber nicht die ganze Komplexität von C++
- Objective-C mit ein bisschen C++ gewürzt
- Einfach zu lernen, auch ohne C++ Erfahrung

Objective-C++

```
if  let document = self.document  {
    [self loadDocument:document]
}
```

Objective-C++

```
#define let const auto

if (let document = self.document) {
    [self loadDocument:document];
}
```

Objective-C++

```
#define let const auto

if (let document = self.document) {
    [self loadDocument:document];
}
```

Objective-C++

```
#define let const auto

if (let document = self.document) {
    [self loadDocument:document];
}
```

Objective-C++

```
void(^handler)(NSData *,
        NSURLResponse *,
              NSError *) = ^(
                 NSData *data,
           NSURLResponse *response,
                 NSError *error) {
  // some handler
};
```

Objective-C++

```
auto handler = ^(NSData *data,
           NSURLResponse *response,
                 NSError *error) {
  // some handler
};
```

Objective-C++

```
std::vector<CGFloat> ratios;
ratios.reserve(sampleSize);
for (int idx = 0; idx < max; idx += max/sample) {
  CGFloat r = <someCalculation>;
  ratios.push_back(r);
}
std::sort(ratios.begin(), ratios.end());
middleRatio = ratios[ratios.size() / 2];
```

Objective-C++

```
std::vector<CGFloat> ratios;
ratios.reserve(sampleSize);
for (int idx = 0; idx < max; idx += max/sample) {
  CGFloat r = <someCalculation>;
  ratios.push_back(r);
}
std::sort(ratios.begin(), ratios.end());
middleRatio = ratios[ratios.size() / 2];
```

Objective-C++

```
std::vector<CGFloat> ratios;
ratios.reserve(sampleSize);
for (int idx = 0; idx < max; idx += max/sample) {
  CGFloat r = <someCalculation>;
  ratios.push_back(r);
}
std::sort(ratios.begin(), ratios.end());
middleRatio = ratios[ratios.size() / 2];
```

Objective-C++

```
std::vector<CGFloat> ratios;
ratios.reserve(sampleSize);
for (int idx = 0; idx < max; idx += max/sample) {
  CGFloat r = <someCalculation>;
  ratios.push_back(r);
}
std::sort(ratios.begin(), ratios.end());
middleRatio = ratios[ratios.size() / 2];
```

Objective-C++

```
std::vector<CGFloat> ratios;
ratios.reserve(sampleSize);
for (int idx = 0; idx < max; idx += max/sample) {
  CGFloat r = <someCalculation>;
  ratios.push_back(r);
}
std::sort(ratios.begin(), ratios.end());
middleRatio = ratios[ratios.size() / 2];
```

Objective-C++

```
std::vector<CGFloat> ratios;
ratios.reserve(sampleSize);
for (int idx = 0; idx < max; idx += max/sample) {
  CGFloat r = <someCalculation>;
  ratios.push_back(r);
}
std::sort(ratios.begin(), ratios.end());
middleRatio = ratios[ratios.size() / 2];
```

Objective-C++

```
std::vector<CGFloat> ratios;
ratios.reserve(sampleSize);
for (int idx = 0; idx < max; idx += max/sample) {
  CGFloat r = <someCalculation>;
  ratios.push_back(r);
}
std::sort(ratios.begin(), ratios.end());
middleRatio = ratios[ratios.size() / 2];
```

Objective-C++

Objective-C	C++
NSArray	std::vector
NSDictionary	std::unordered_map
NSSet	std::unordered_set
NSOrderedSet	std::set
	std::map

Objective-C++

- Objective-C und C++ Objekte können beliebig vermischt werden
- Objective-C collections für Objective-C Objekte und C++ collections für C++ Objekte und elementare Datentypen

Objective-C++

```
CGSize operator/(const CGSize &lhs, CGFloat f) {
    return CGSize{ lhs.width / f, lhs.height / f };
}

CGSize zoomSize = self.bounds.size / zoomScale;
```

Objective-C++

```
CGSize operator/(const CGSize &lhs, CGFloat f) {
    return CGSize{ lhs.width / f, lhs.height / f };
}

CGSize zoomSize = self.bounds.size / zoomScale;
```

Objective-C++

```
@interface MyObject () {
  std::mutex _myLock;
}
@end

@implementation MyObject
- (void)doSomething {
  std::lock_guard<std::mutex> lock(_myLock);
  // do stuff
}
@end
```

Objective-C++

```
@interface MyObject () {
  std::mutex _myLock;
}
@end

@implementation MyObject
- (void)doSomething {
  std::lock_guard<std::mutex> lock(_myLock);
  // do stuff
}
@end
```

Objective-C++

```
@interface MyObject () {
  std::mutex _myLock;
}
@end

@implementation MyObject
- (void)doSomething {
  std::lock_guard<std::mutex> lock(_myLock);
  // do stuff
}
@end
```

Objective-C++

```
@interface MyObject () {
  std::mutex _myLock;
}
@end

@implementation MyObject
- (void)doSomething {
  // do stuff without lock
  {
    std::lock_guard<std::mutex> lock(_myLock);
    // do stuff that requires locking
  }
  // continue without lock
}
@end
```

Demo

Objective-C++

- Kompilieren dauert 2-3x länger wenn alle Dateien Objective-C++ sind
- Es gibt mehr Warnungen, insbesondere bei impliziten Casts
- Kein Refactoring in Xcode für Objective-C++

Objective-C++

- Achtung: Bei C++ können auch Klassen value types sein!
- C++ sollte nicht in Headern verwendet werden
  - Kann dann nur noch in andere Objective-C++ Dateien importiert werden
  - Funktioniert nicht mit Swift
  - Wenn nicht vermeidbar: `#ifdef __cplusplus`

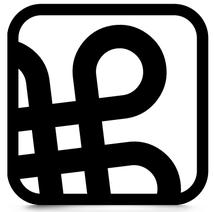
Zusammenfassung

Zusammenfassung

- Swift hat eine menge Potential - aber ist noch nicht so weit
- Es gibt eine menge Dinge die man bei Swift beachten muss
- Eine Menge der neuen Features von Swift sind gar nicht so neu und können in anderen Sprachen schon länger benutzt werden

Danke

Michael Ochs
@_mochs
https://pspdfkit.com/blog/



Macoun