

Advances in Objective-C

Session 404

Doug Gregor

Senior Engineer, LLVM Frontend Team

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

Objective-C is on the Move

TIOBE Programming Community Index, May 2013

Programming Language	
1	C
2	Java
3	C++
4	Objective-C

Objective-C is on the Move

TIOBE Programming Community Index, May 201

Programming Language	
1	C
2	Java
3	Objective-C
4	C++

Developer
Productivity

Software
Quality

Developer Productivity

- Eliminating boilerplate
- Simplifying common operations
- Providing great tools

Software Quality

Developer Productivity

- Eliminating boilerplate
- Simplifying common operations
- Providing great tools

Software Quality

- Catching bugs early
- Automating error-prone tasks
- Encouraging best practices

Roadmap

- Modules
- Better productivity
- ARC improvements

Modules

Frameworks at the Core

Building blocks of apps



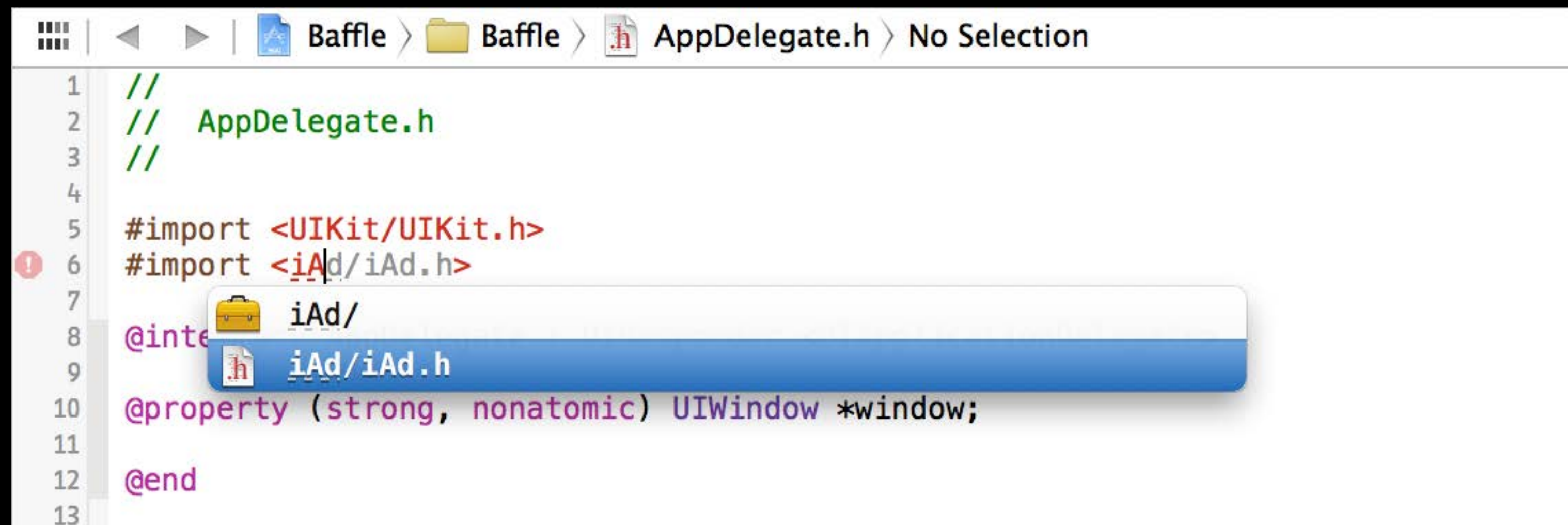
Frameworks at the Core

Building blocks of apps



Using a Framework

Import the framework...

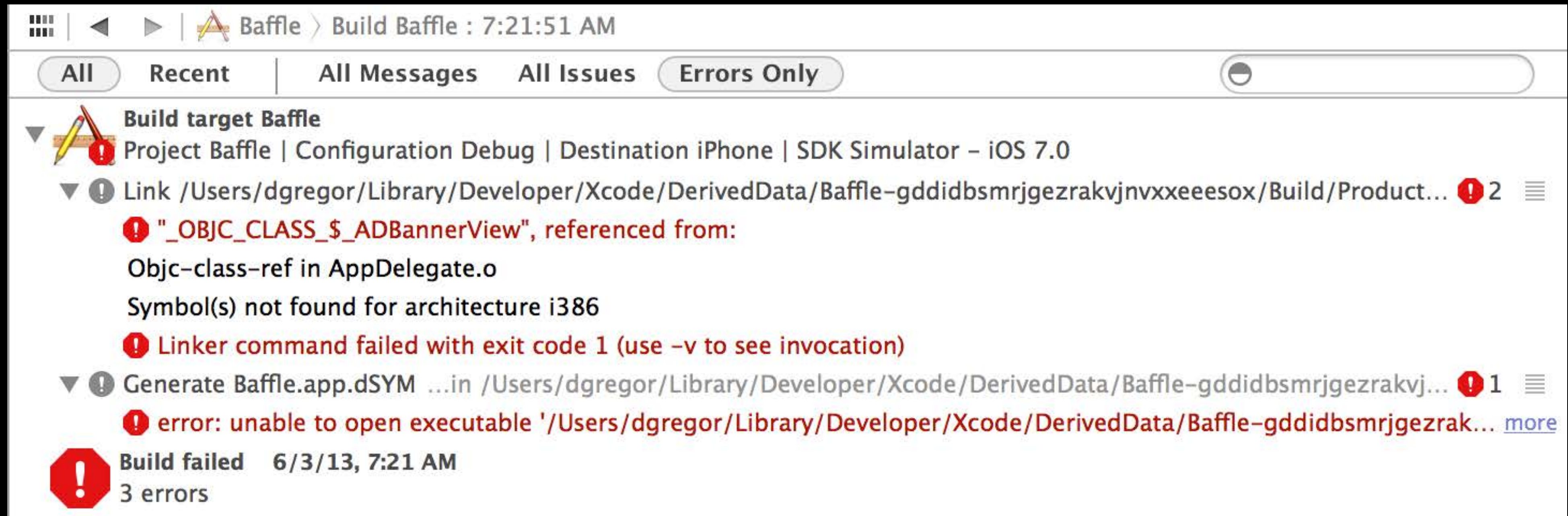


The screenshot shows the Xcode IDE with the AppDelegate.h file open. The code contains the following lines:

```
1 //  
2 // AppDelegate.h  
3 //  
4  
5 #import <UIKit/UIKit.h>  
6 #import <iAd/iAd.h>  
7  
8 @interface AppDelegate : UIResponder {  
9  
10 @property (strong, nonatomic) UIWindow *window;  
11  
12 @end
```

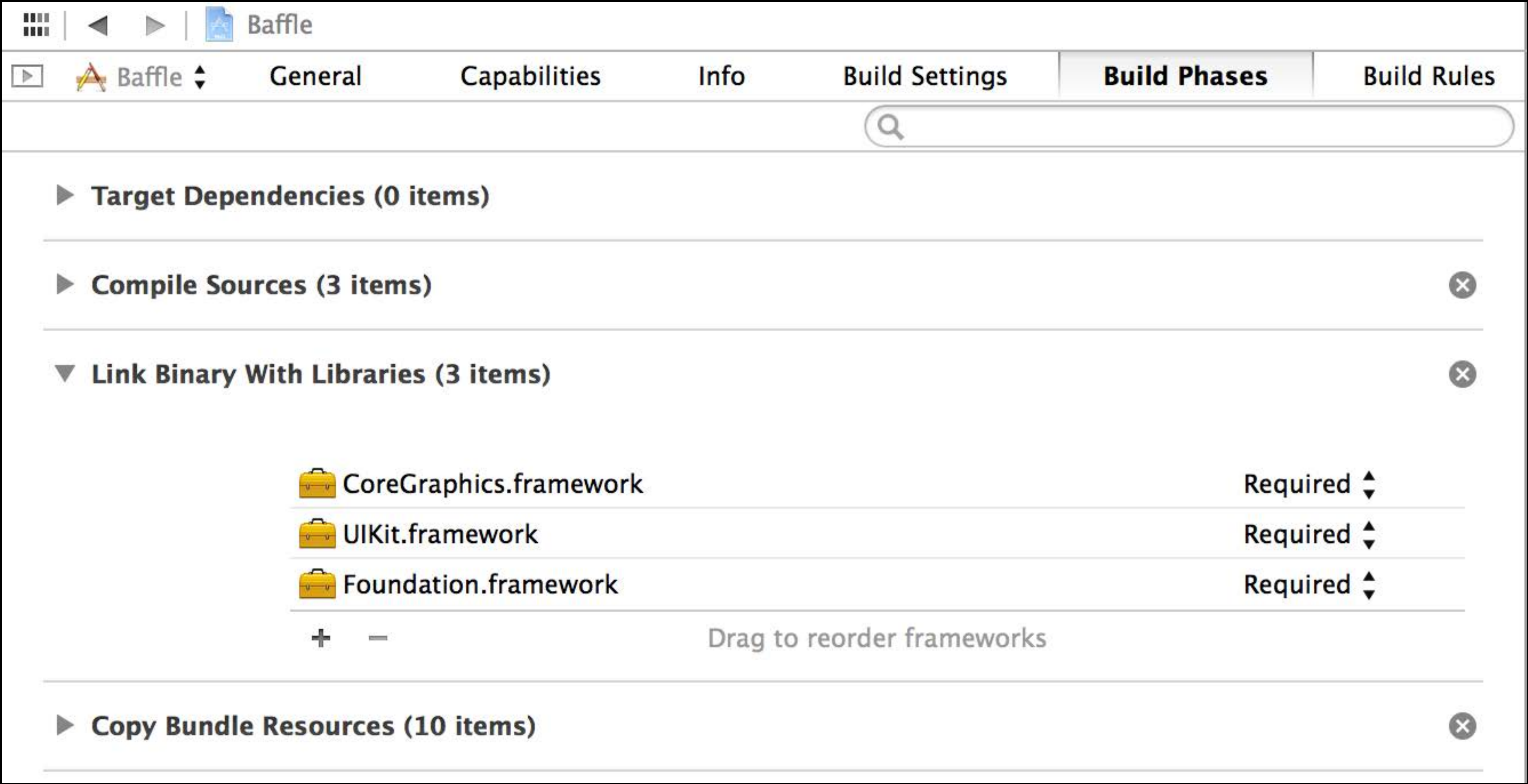
A red exclamation mark icon is visible next to line 6. A file selection popup is displayed over line 6, showing the path `iAd/iAd.h` selected.

Using a Framework



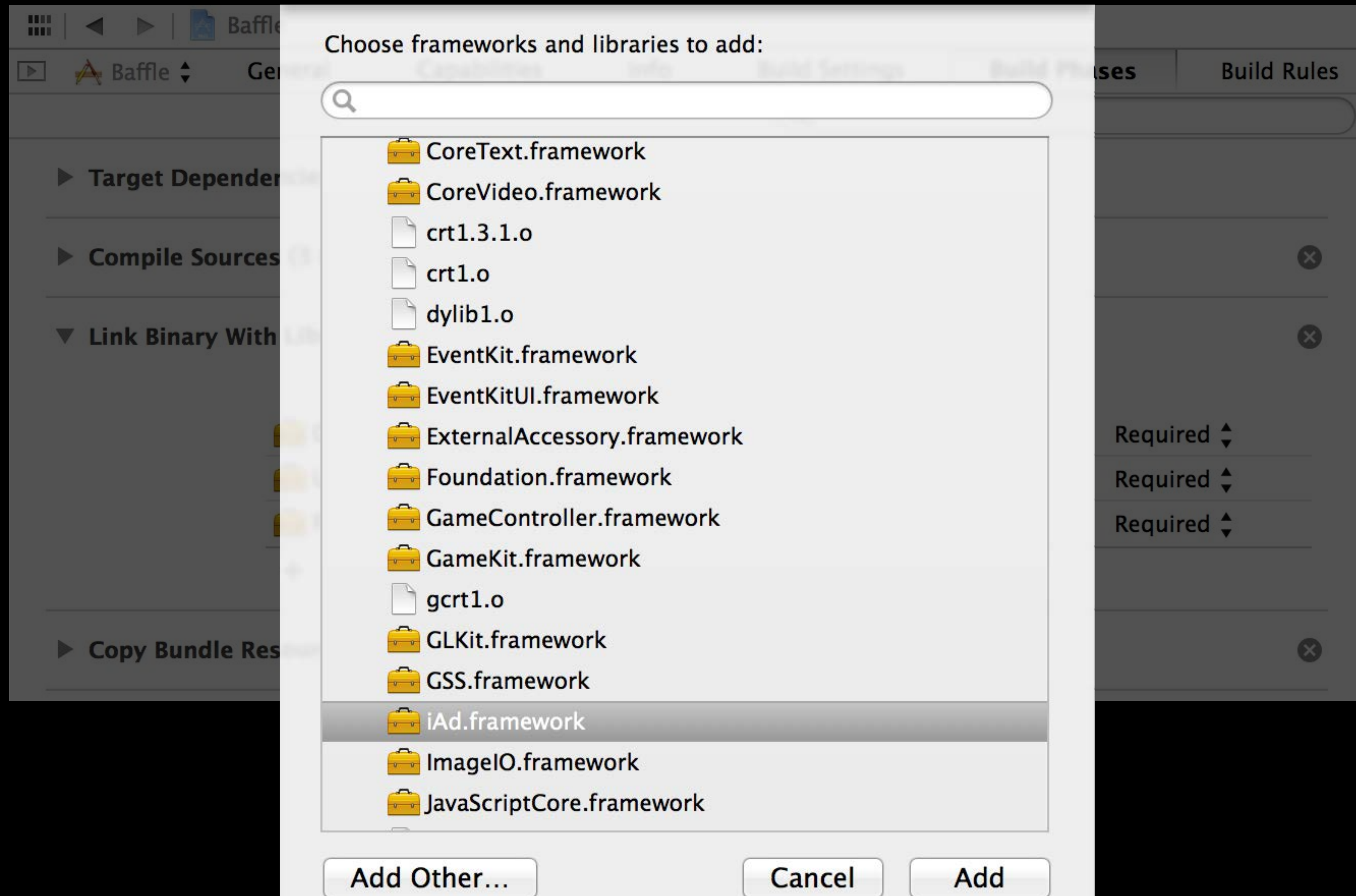
Using a Framework

Import and link against the framework



Using a Framework

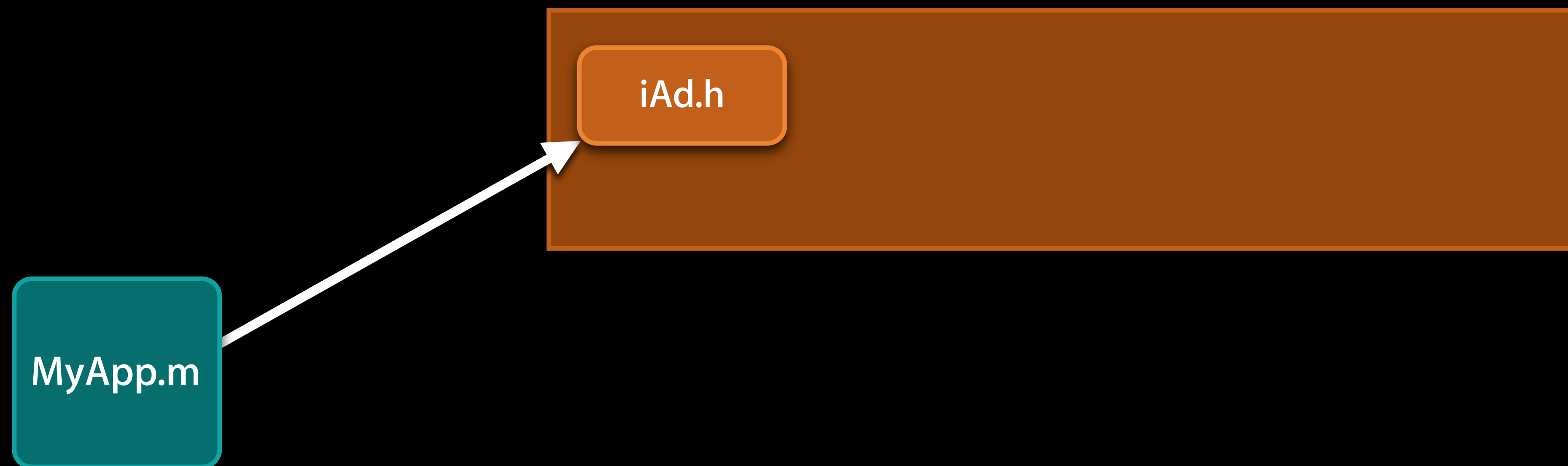
Import and link against the framework



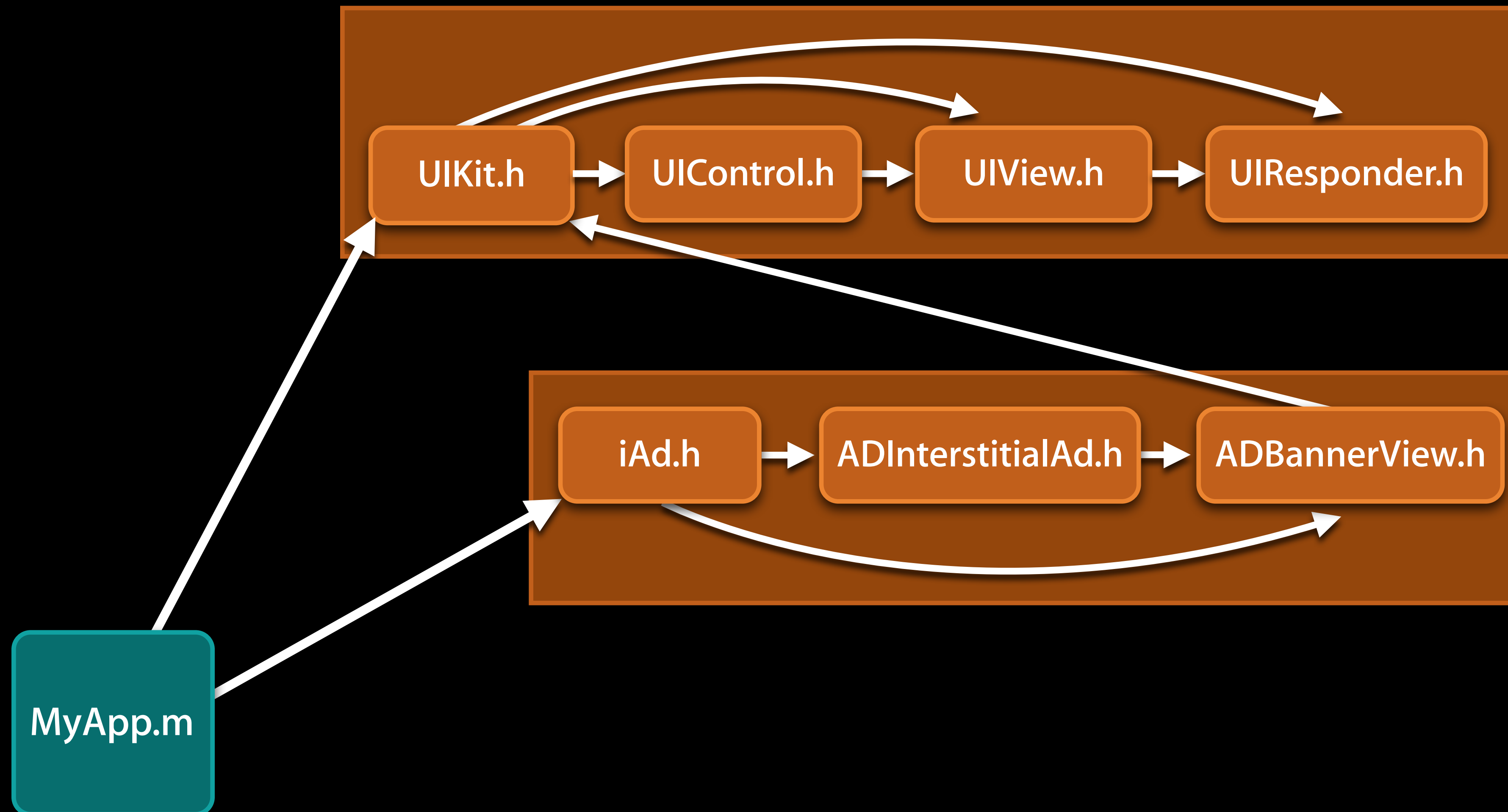
Using a Framework



Headers and Frameworks



Headers and Frameworks



Preprocessing and Textual Inclusion

```
#import <iAd/iAd.h>  
@implementation AppDelegate  
// ...  
@end
```

Preprocessing and Textual Inclusion

```
#import <iAd/iAd.h>
@implementation AppDelegate
// ...
@end
```

```
/* iAd/iAd.h */
#import <iAd/ADBannerView.h>
#import <iAd/ADBannerView_Deprecated.h>
#import <iAd/ADInterstitialAd.h>
```

Preprocessing and Textual Inclusion

```
#import <iAd/ADBannerView.h>
#import <iAd/ADBannerView_Deprecated.h>
#import <iAd/ADInterstitialAd.h>

@implementation AppDelegate
// ...
@end
```

Preprocessing and Textual Inclusion

```
#import <iAd/ADBannerView.h>
#import <iAd/ADBannerView_Deprecated.h>
#import <iAd/ADInterstitialAd.h>

@implementation AppDelegate
// ...
@end
```

```
/* iAd/ADBannerView.h */
@interface ADBannerView : UIView
@property (nonatomic,readonly) ADAdType adType;

- (id)initWithAdType:(ADAdType)type

/* ... */

@end
```

Preprocessing and Textual Inclusion

```
@interface ADBannerView : UIView
@property (nonatomic,readonly) ADAdType adType;

- (id)initWithAdType:(ADAdType)type

/* ... */

@end
#import <iAd/ADBannerView_Deprecated.h>
#import <iAd/ADInterstitialAd.h>

@implementation AppDelegate
// ...

@end
```

Preprocessing and Textual Inclusion

```
@interface ADBannerView : UIView
@property (nonatomic,readonly) ADAdType adType;

- (id)initWithAdType:(ADAdType)type

/* ... */

@end
#import <iAd/ADBannerView_Deprecated.h>
#import <iAd/ADInterstitialAd.h>

@implementation AppDelegate
// ...
@end
```

```
/*iAd/ADBannerView_Deprecated.h*/
```

```
/* iAd/ADInterstitialAd.h */
```


Preprocessing and Textual Inclusion

```
@interface ADBannerView : UIView
@property (nonatomic,readonly) ADAdType adType;

- (id)initWithAdType:(ADAdType)type

/* ... */

@end

@implementation AppDelegate
// ...

@end
```

Header Fragility

```
#define readonly 0x01
#import <iAd/iAd.h>

@implementation AppDelegate
// ...
@end
```

Header Fragility

```
@interface ADBannerView : UIView
@property (nonatomic,0x01) ADAdType adType;

- (id)initWithAdType:(ADAdType)type

/* ... */

@end

@implementation AppDelegate
// ...
@end
```

Header Fragility

```
! @interface ADBannerView : UIView
  @property (nonatomic, 0x01) ADAdType adType;

- (id)initWithAdType:(ADAdType)type

/* ... */

@end

@implementation AppDelegate
// ...

@end
```

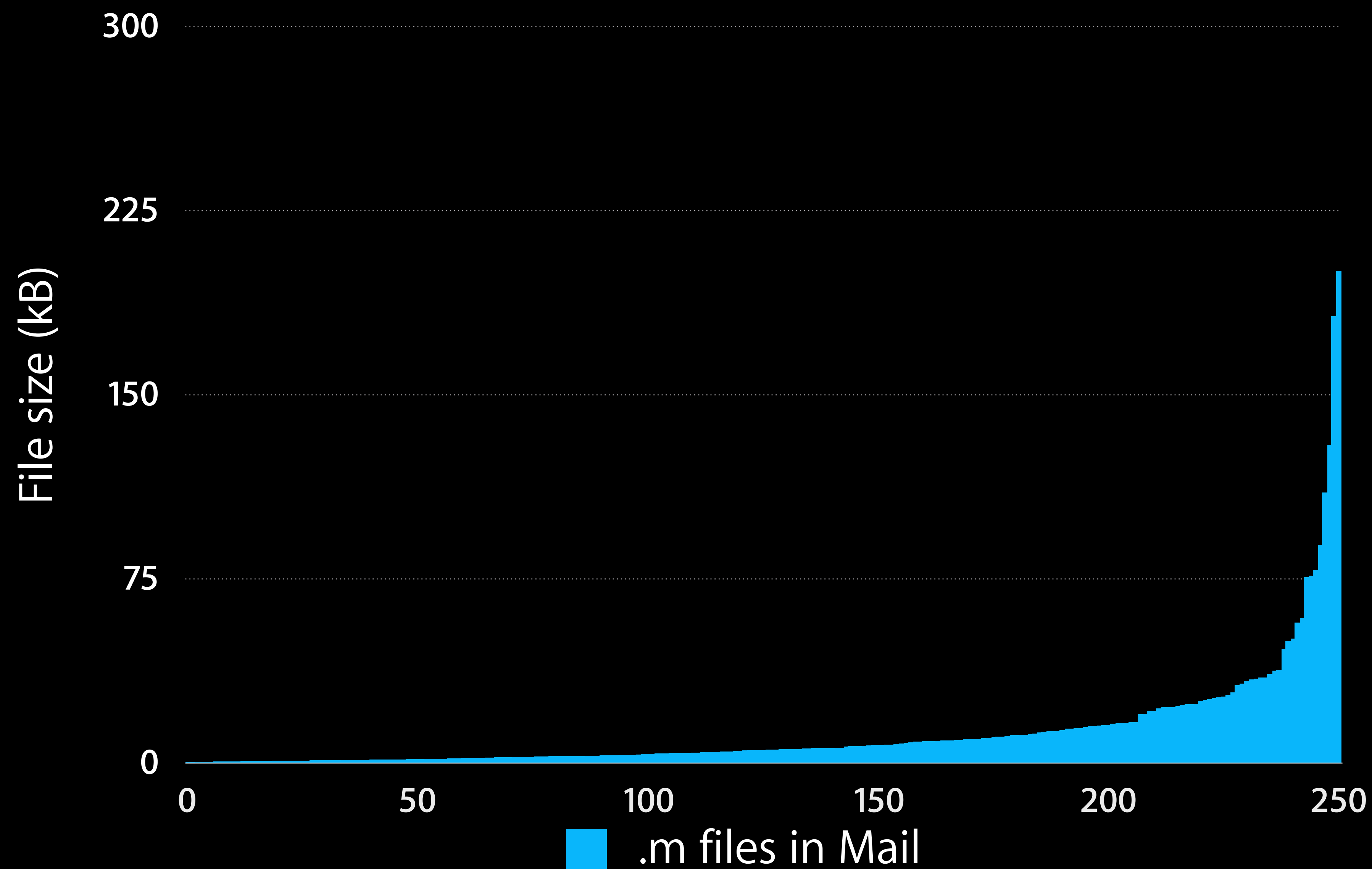
Header Fragility

```
! @interface ADBannerView : UIView
  @property (nonatomic, 0x01) ADAdType adType;
- (id)initWithAdType:(ADAdType)type
/* ... */
@end

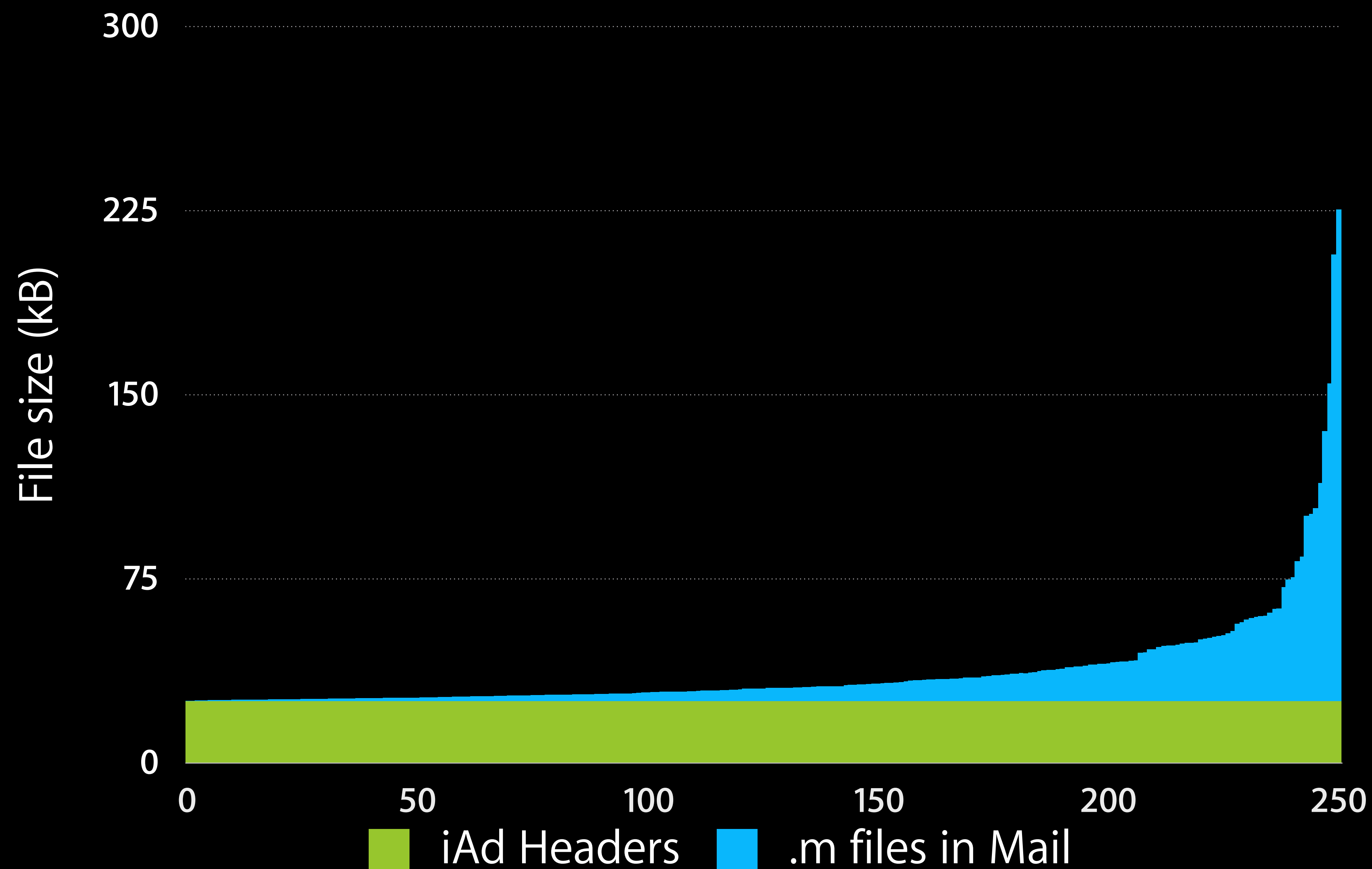
@implementation AppDelegate
// ...
@end
```

- UPPERCASE_MACRO_NAMES
- Manifests as header ordering problems

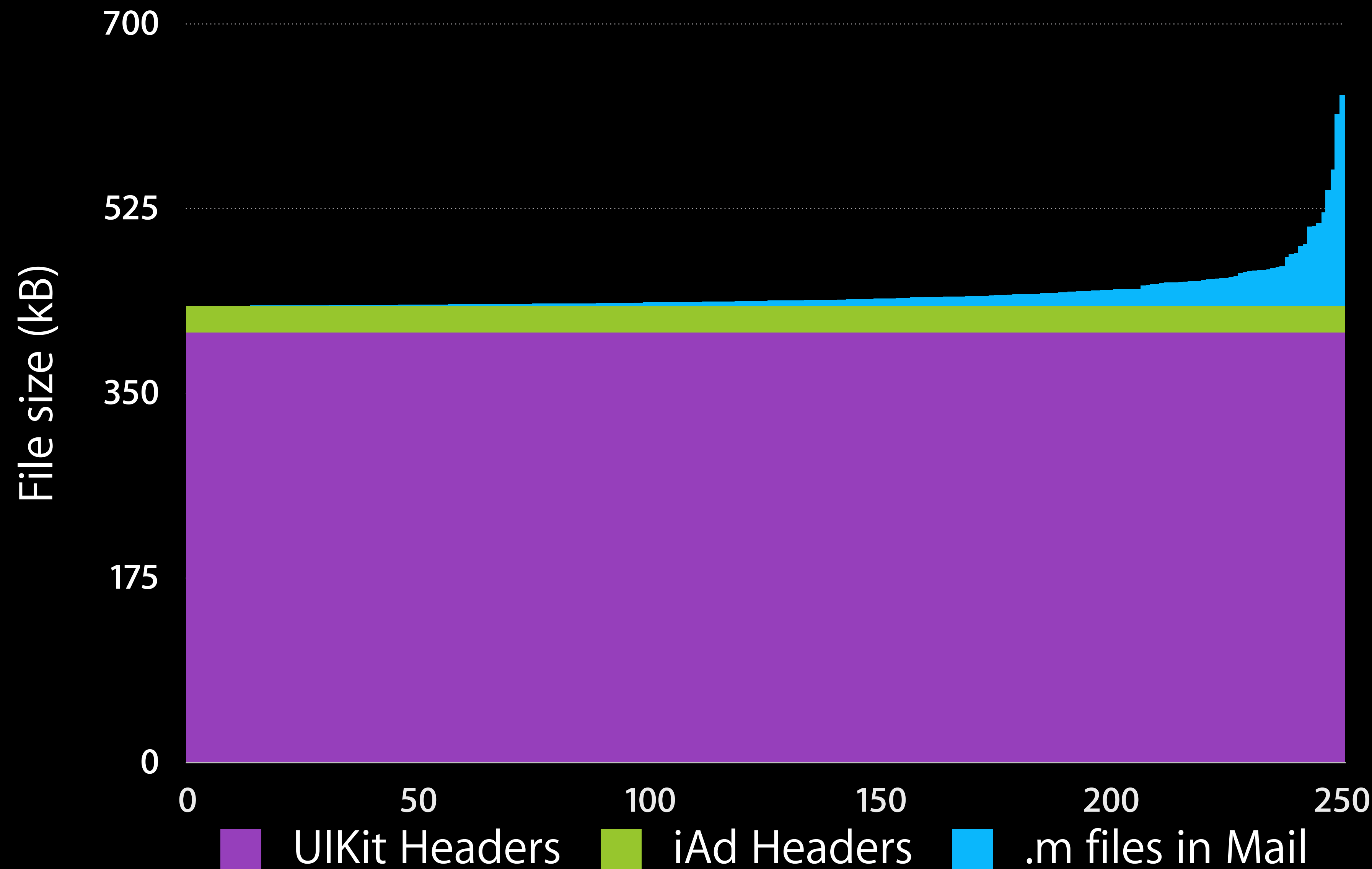
Inherently Non-Scalable Compile Times



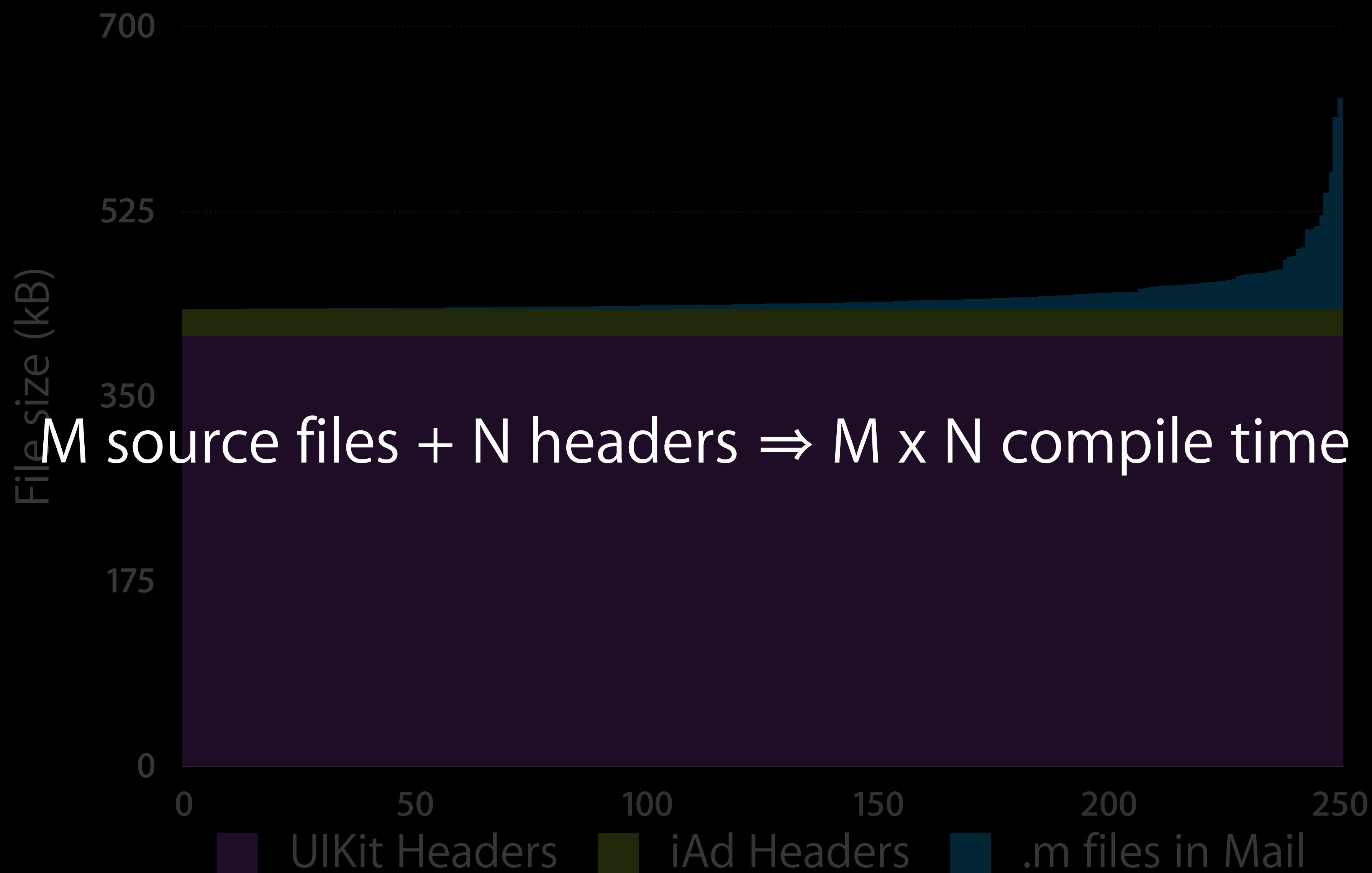
Inherently Non-Scalable Compile Times



Inherently Non-Scalable Compile Times

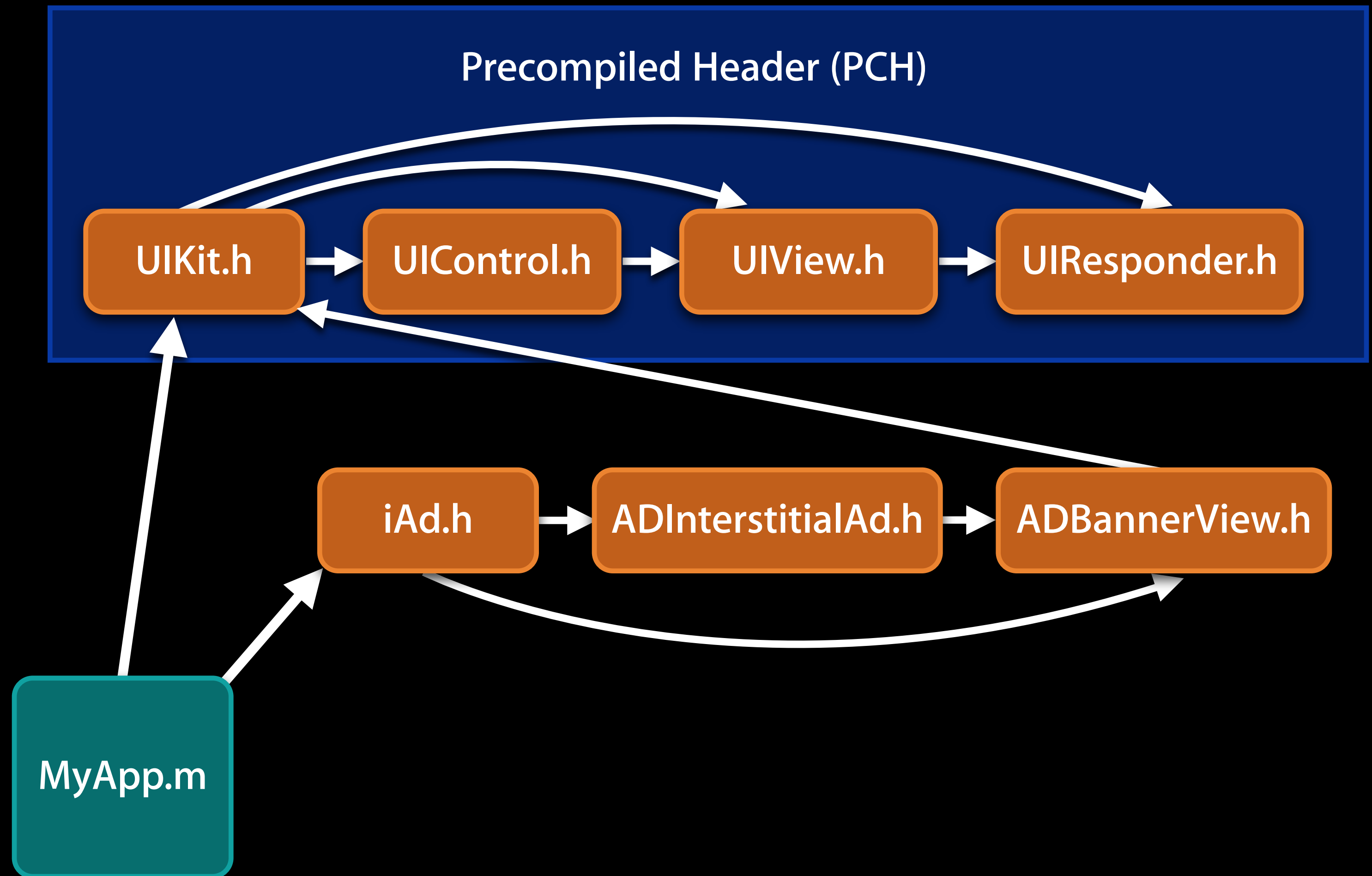


Inherently Non-Scalable Compile Times



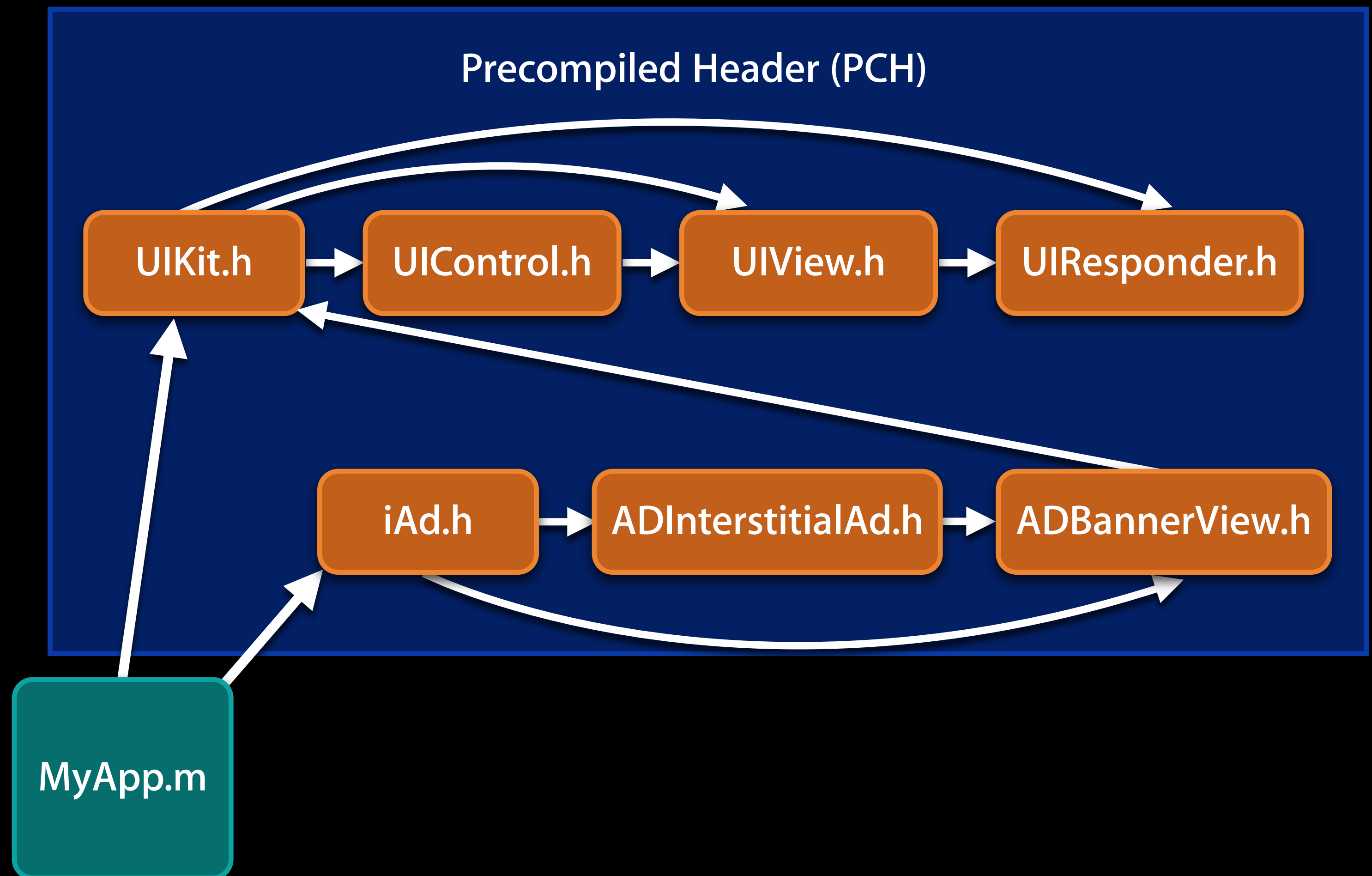
What About Precompiled Headers?

- Precompiled headers help significantly
 - UIKit / Cocoa come for free



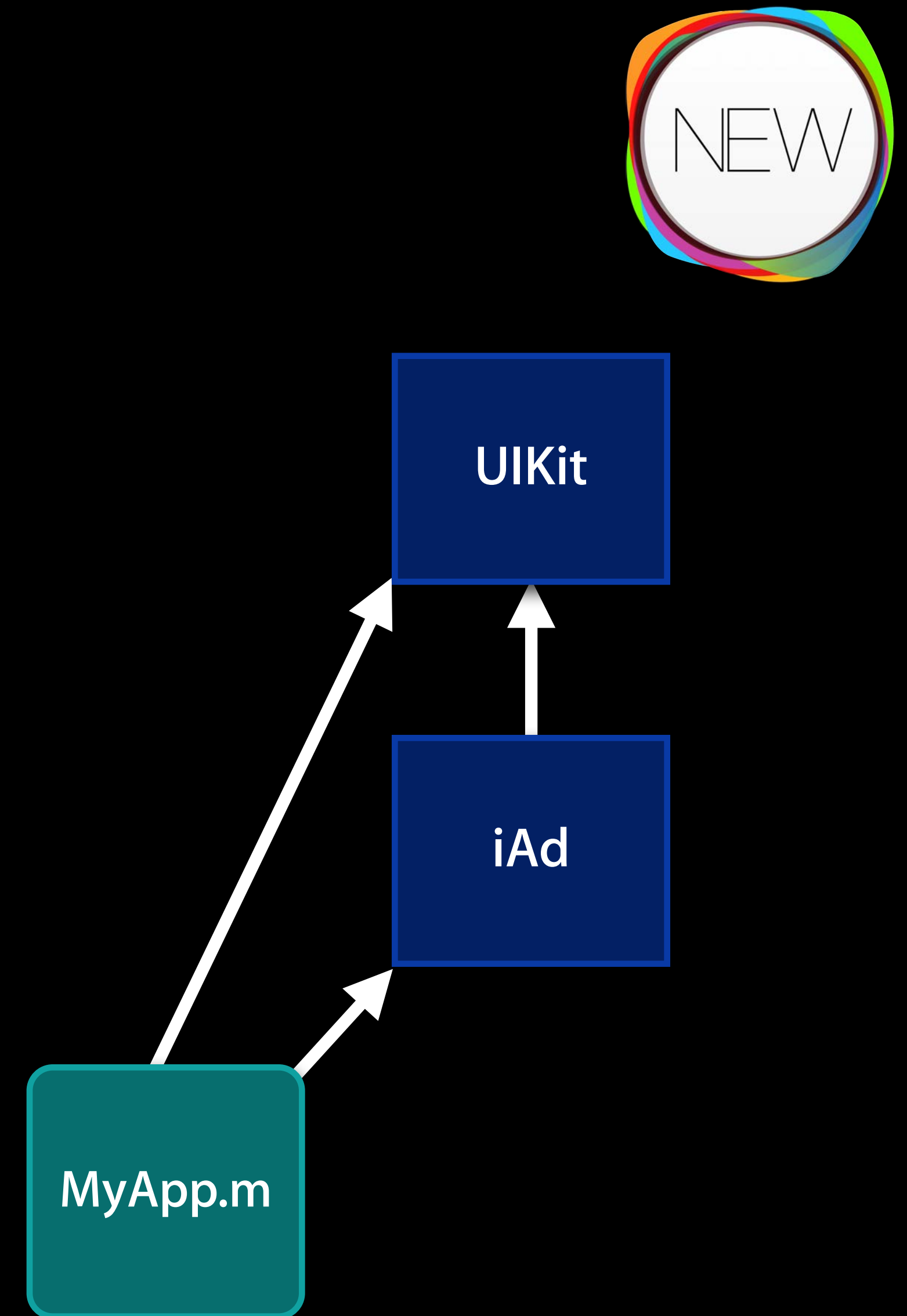
What About Precompiled Headers?

- Precompiled headers help significantly
 - UIKit / Cocoa come for free
- Maintenance burden
- Namespace pollution



Modules

- Modules encapsulate a framework
 - Interface (API)
 - Implementation (dylib)
- Separately compiled

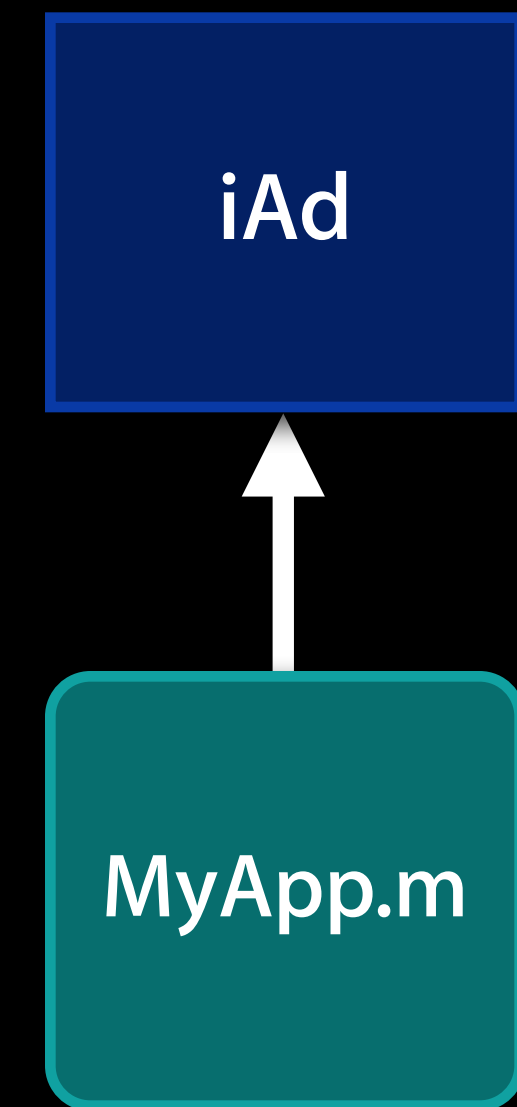


Semantic Import

- New `@import` declaration accesses framework API

```
@import iAd;
```

- Imports complete semantic description of a framework
 - Doesn't need to parse the headers
 - Local macro definitions have no effect on framework API

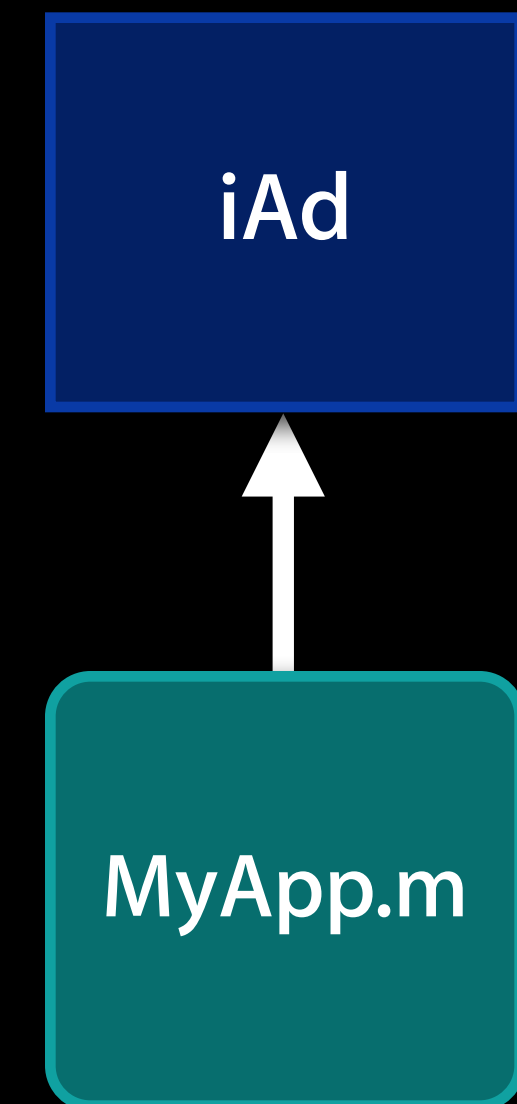


Semantic Import

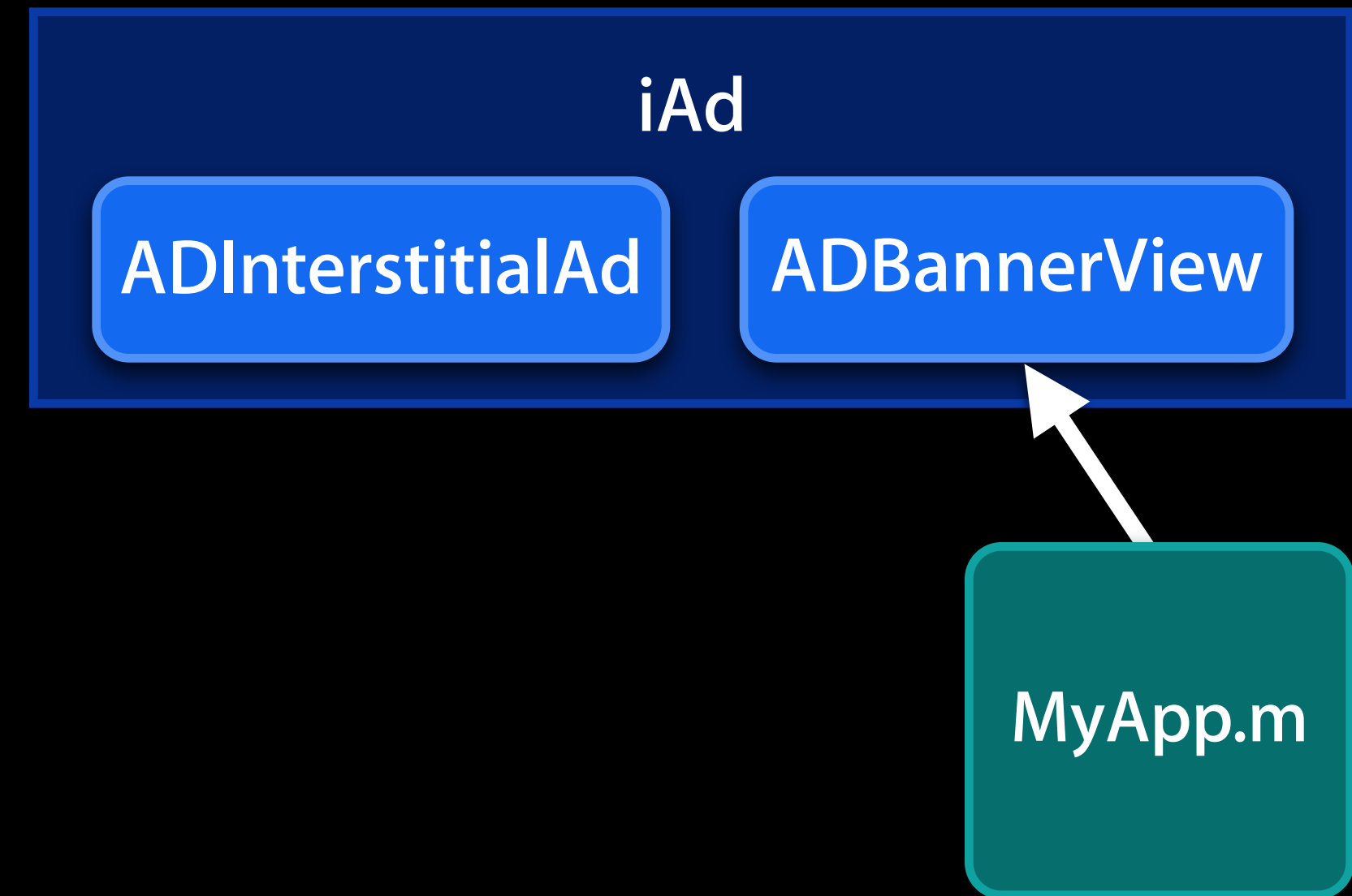
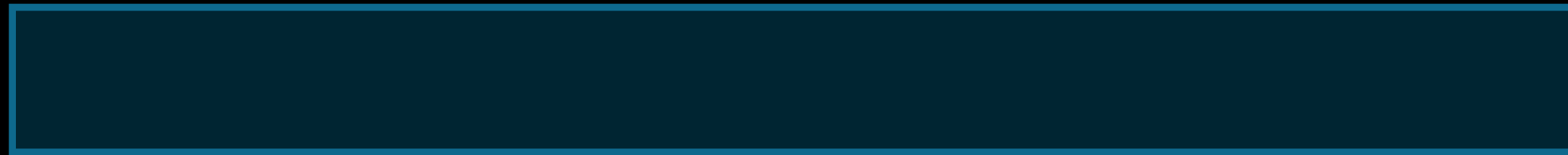
- New `@import` declaration accesses framework API

```
#define readonly 0x01  
@import iAd;
```

- Imports complete semantic description of a framework
 - Doesn't need to parse the headers
 - Local macro definitions have no effect on framework API



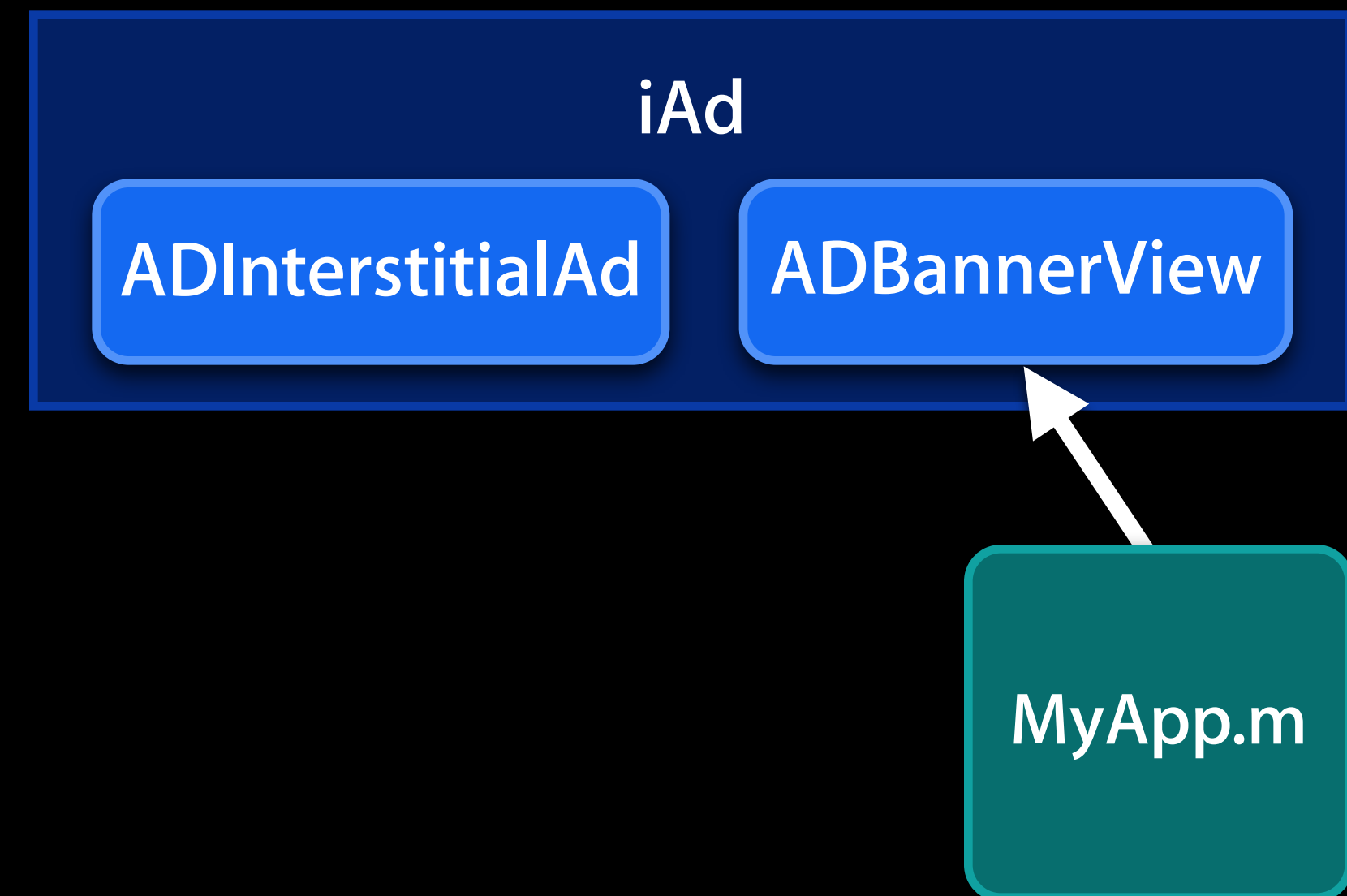
Selective Import



Selective Import

- Import part of a framework

```
@import iAd.ADBannerView;
```



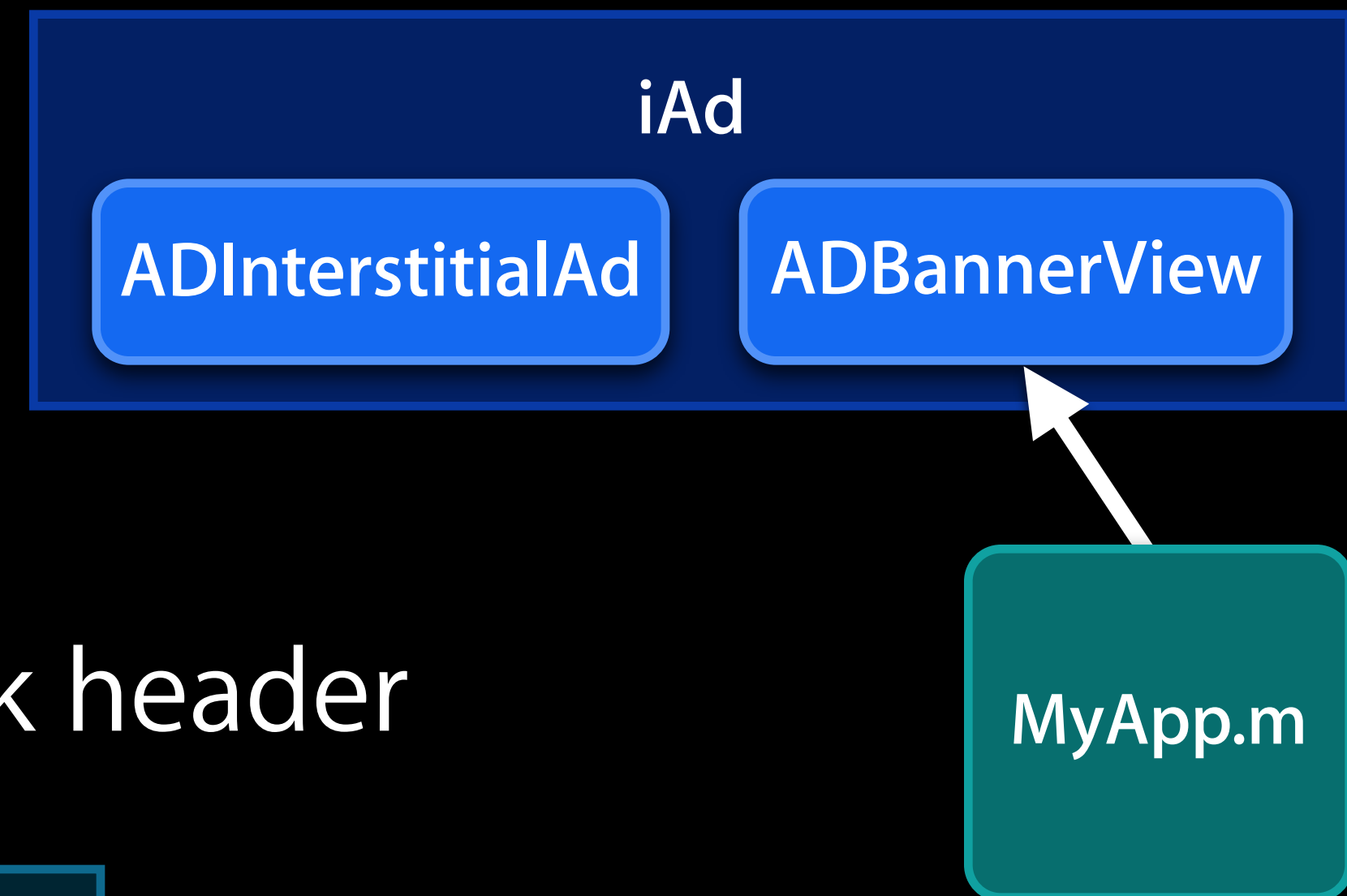
Selective Import

- Import part of a framework

```
@import iAd.ADBannerView;
```

- Similar to importing a specific framework header

```
#import <iAd/ADBannerView.h>
```



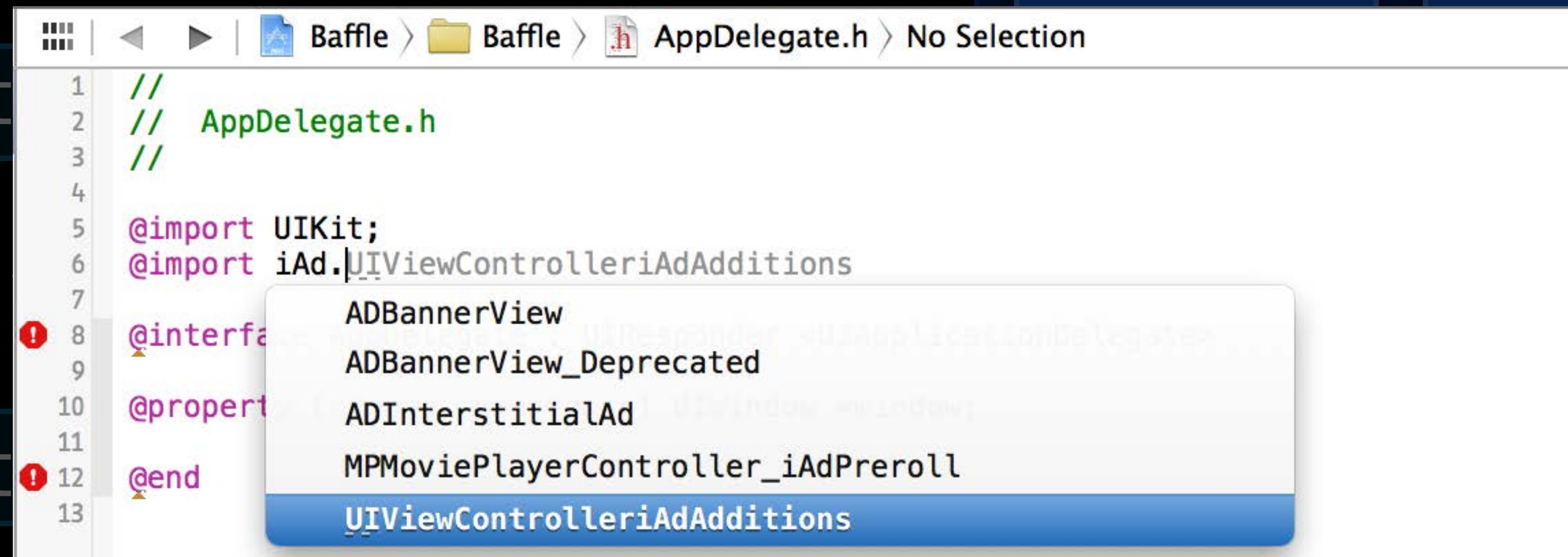
Selective Import

- Import part of a framework

@import

- Similar to

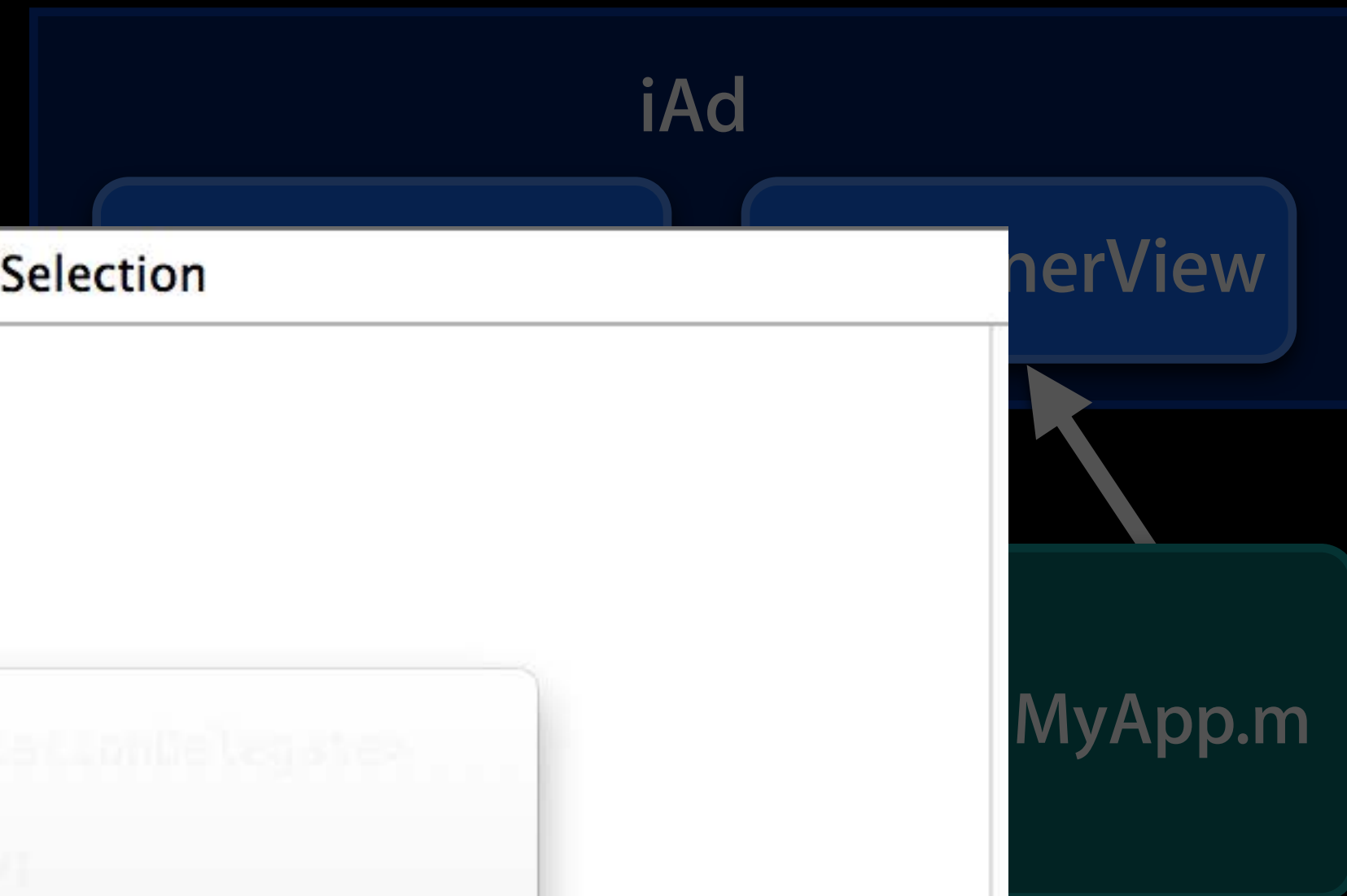
#import



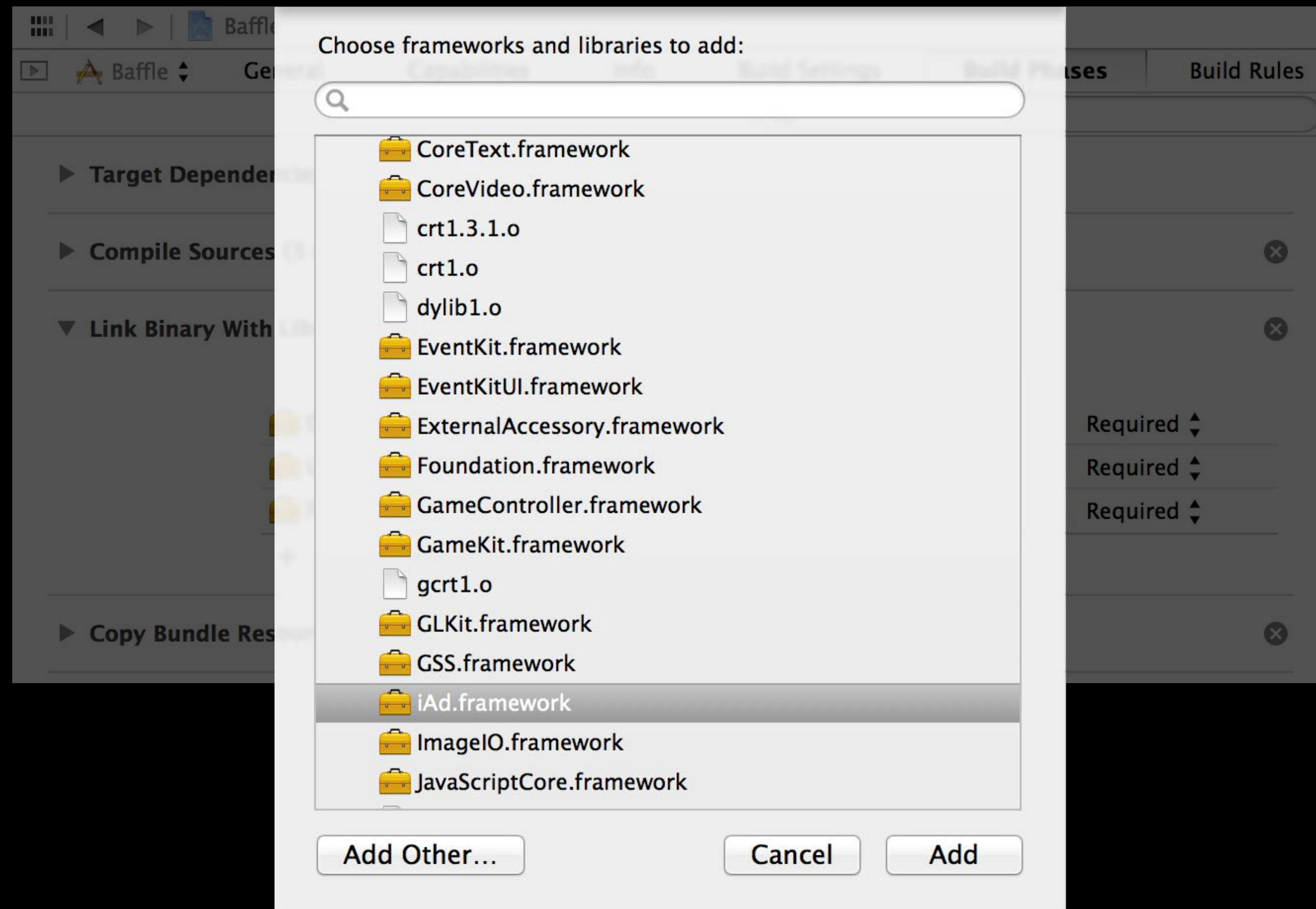
```
1 //
2 // AppDelegate.h
3 //
4
5 @import UIKit;
6 @import iAd.UIViewControlleriAdAdditions
7
8 @interface AppDelegate <UIApplicationDelegate>
9
10 @property (nonatomic, strong) UIWindow *window;
11
12 @end
13
```

A dropdown menu is open over line 6, showing the following options:

- ADBannerView
- ADBannerView_Deprecated
- ADInterstitialAd
- MPMoviePlayerController_iAdPreroll
- UIViewControlleriAdAdditions** (highlighted)

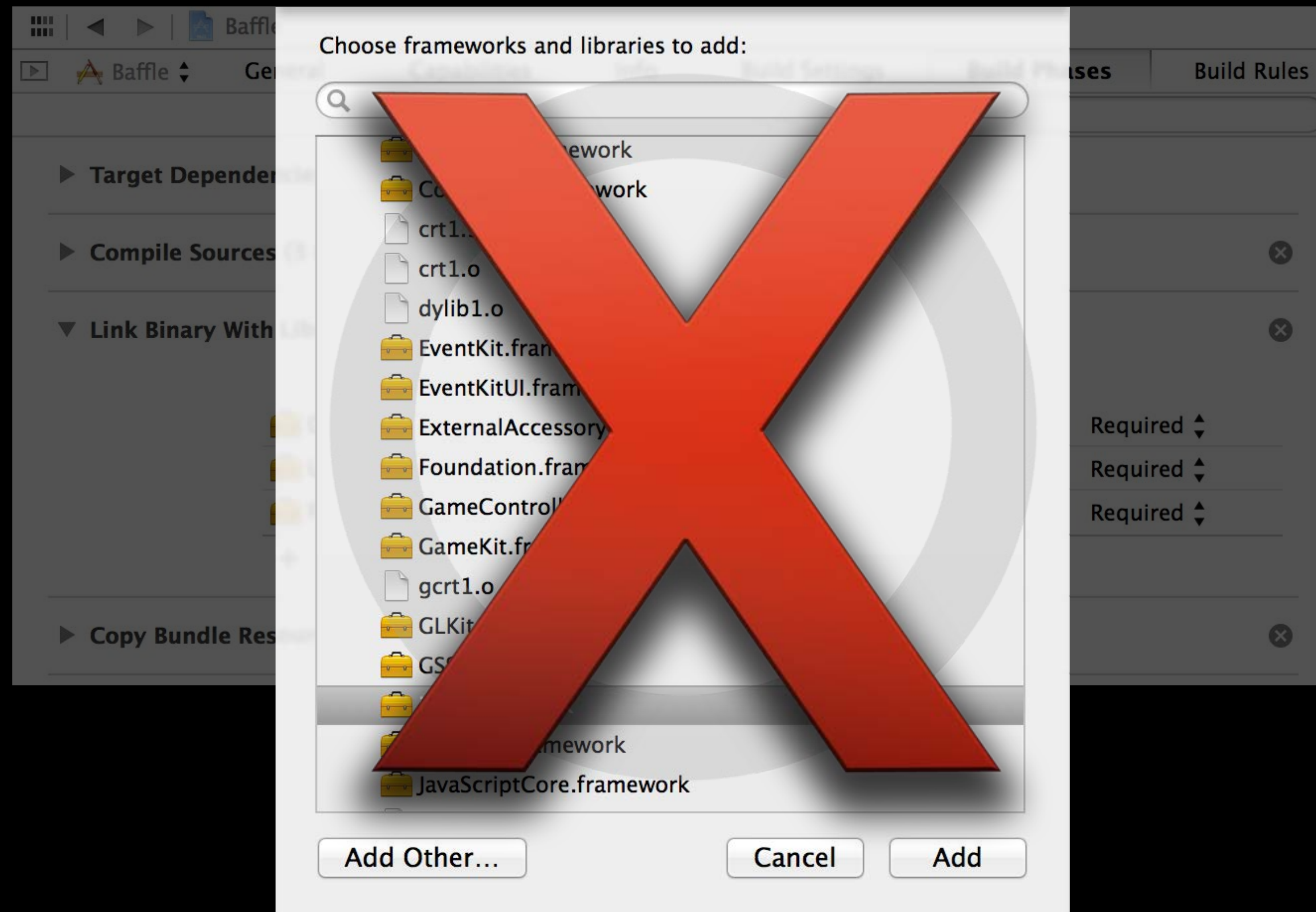


Autolinking



Autolinking

- Eliminates the need to “link binary with libraries”



Using Modules

- Opt in via build settings

Using Modules

- Opt in via build settings
- `#import` and `#include` automatically mapped to `@import`

<code>#import <UIKit/UIKit.h></code>	→	<code>@import UIKit;</code>
<code>#import <iAD/ADBannerView.h></code>	→	<code>@import iAd.ADBannerView;</code>

Using Modules

- Opt in via build settings
- `#import` and `#include` automatically mapped to `@import`

<code>#import <UIKit/UIKit.h></code>	→	<code>@import UIKit;</code>
<code>#import <iAD/ADBannerView.h></code>	→	<code>@import iAd.ADBannerView;</code>

- No source changes required

Using Modules

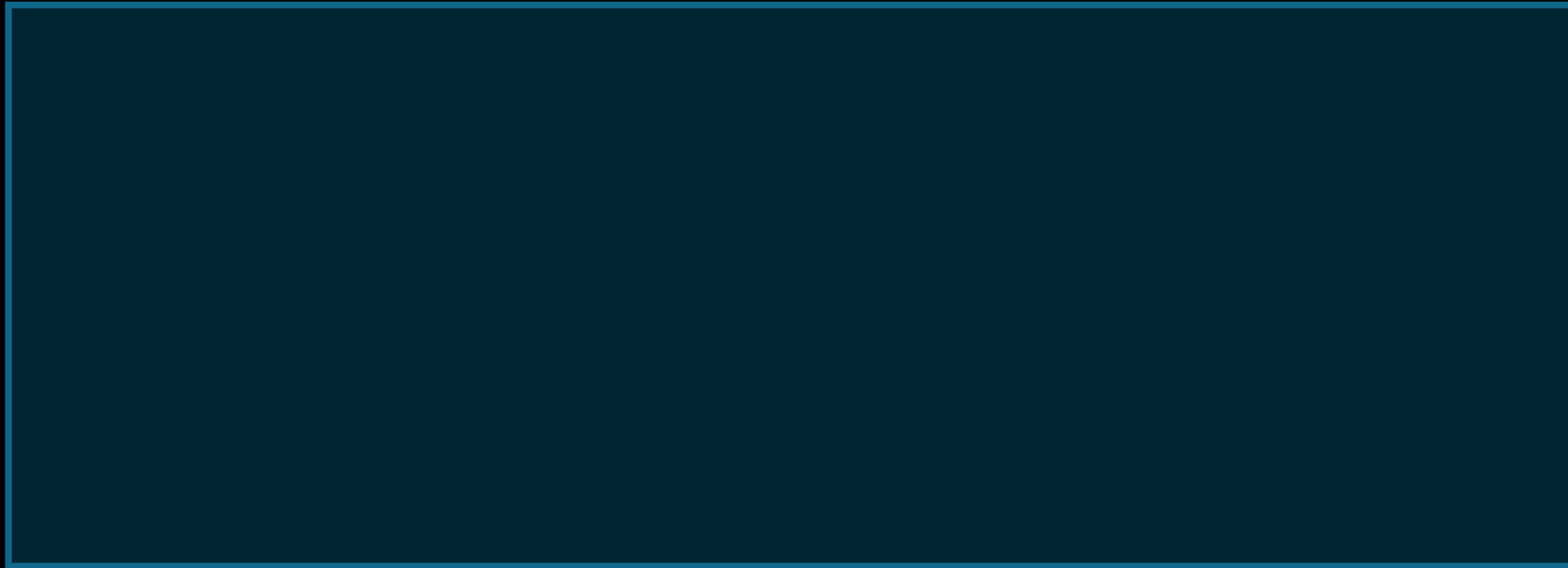
- Opt in via build settings
- `#import` and `#include` automatically mapped to `@import`

<code>#import <UIKit/UIKit.h></code>	→	<code>@import UIKit;</code>
<code>#import <iAD/ADBannerView.h></code>	→	<code>@import iAd.ADBannerView;</code>

- No source changes required
- System frameworks available as modules with iOS 7 / OS X 10.9 SDK

Module Maps

A quick peek under the hood



Module Maps

A quick peek under the hood

- Module maps establish relationship between headers and modules:

```
framework module UIKit {  
    umbrella header "UIKit.h"  
    module * { export * }  
    link framework "UIKit"  
}
```

Module Maps

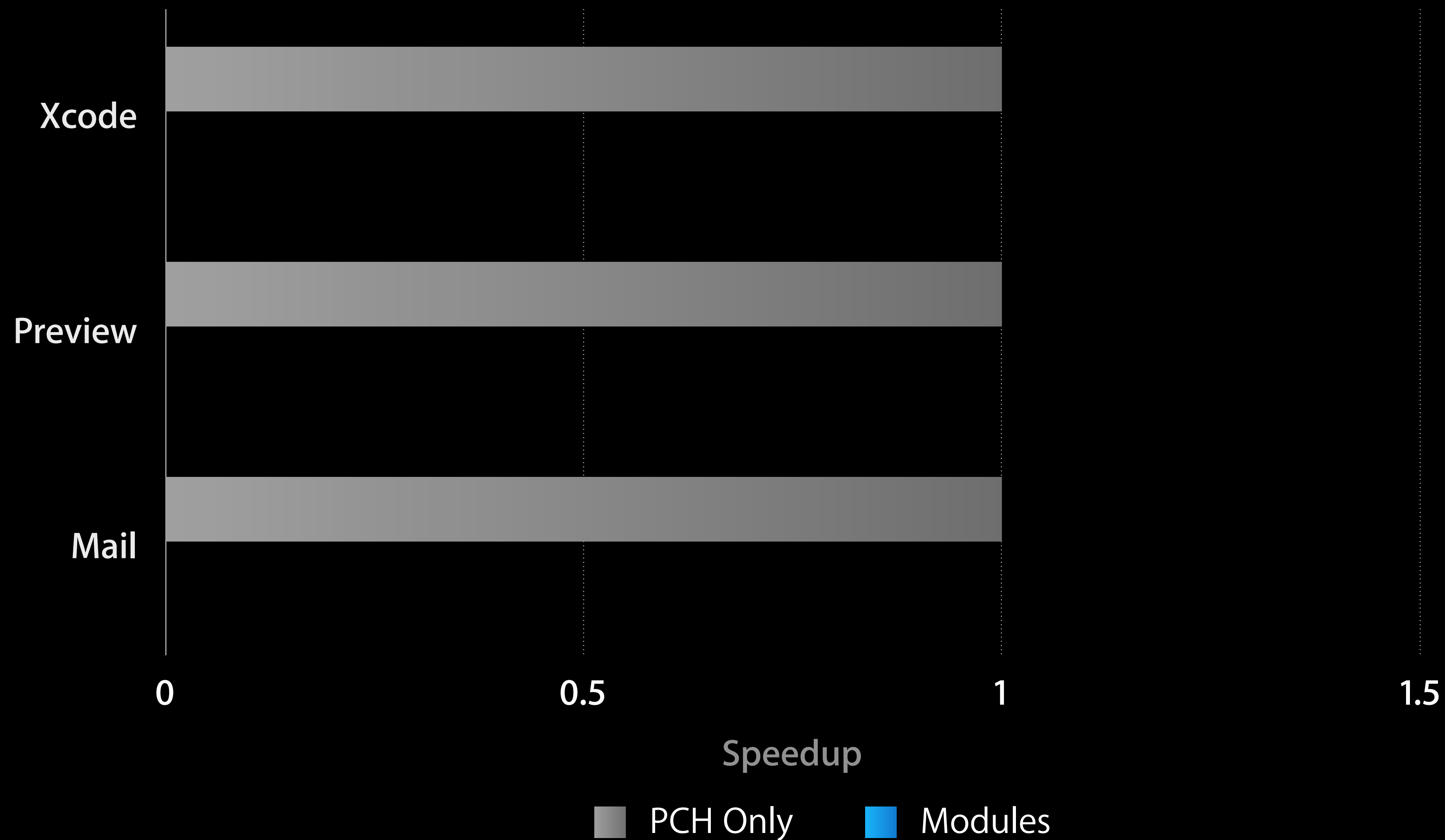
A quick peek under the hood

- Module maps establish relationship between headers and modules:

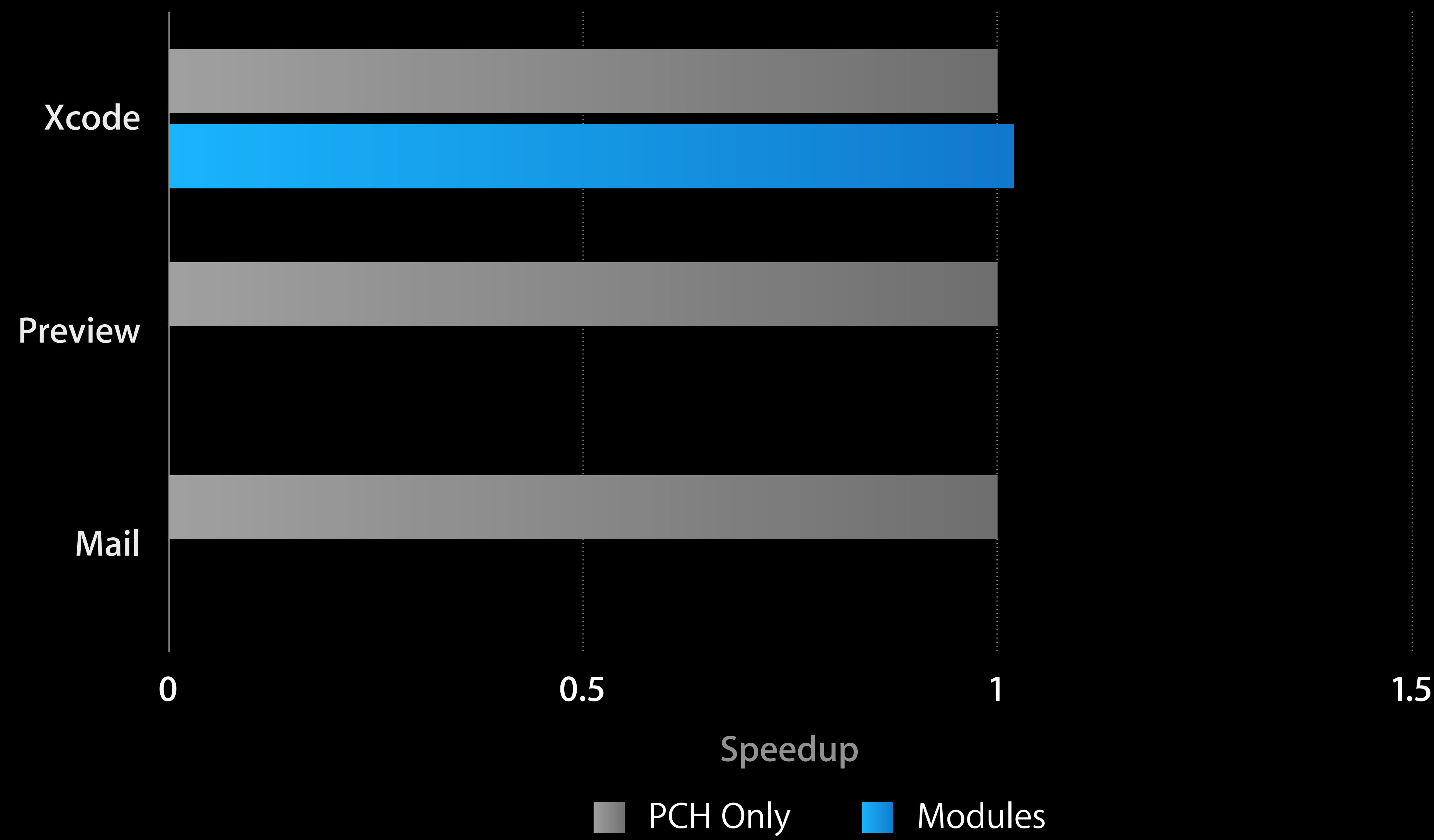
```
framework module UIKit {  
    umbrella header "UIKit.h"  
    module * { export * }  
    link framework "UIKit"  
}
```

- Modules automatically built from headers

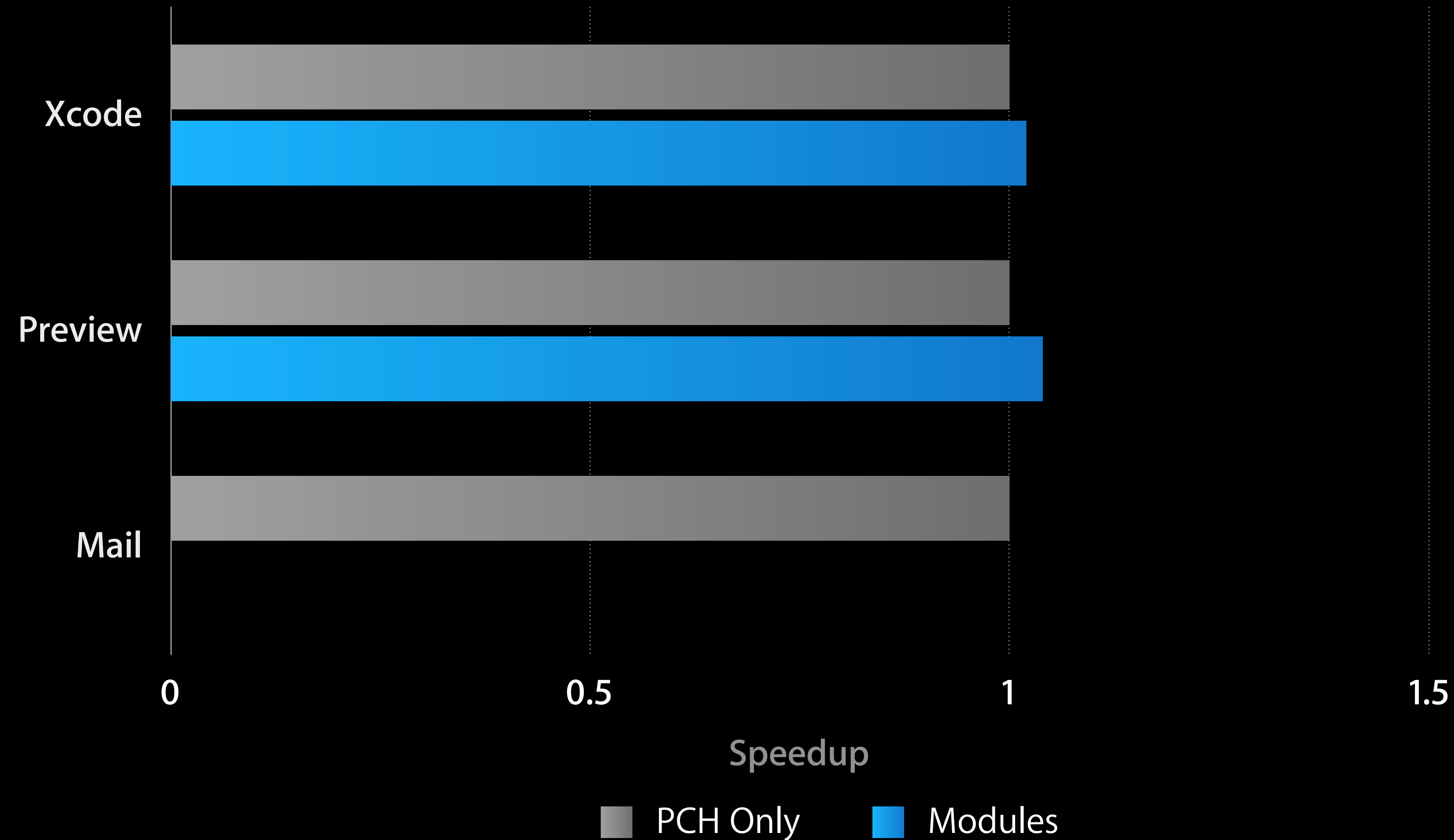
Build Time Improvements



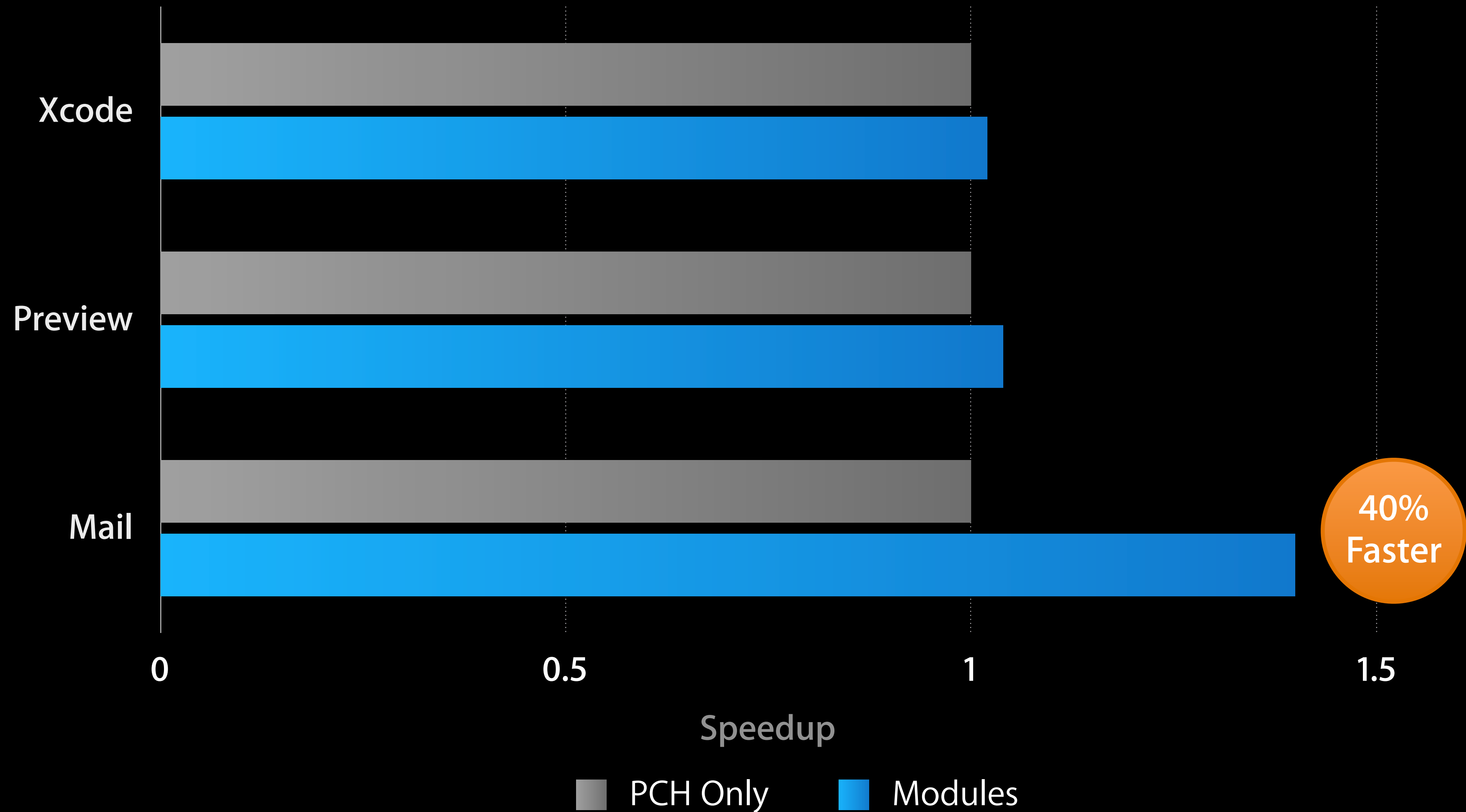
Build Time Improvements



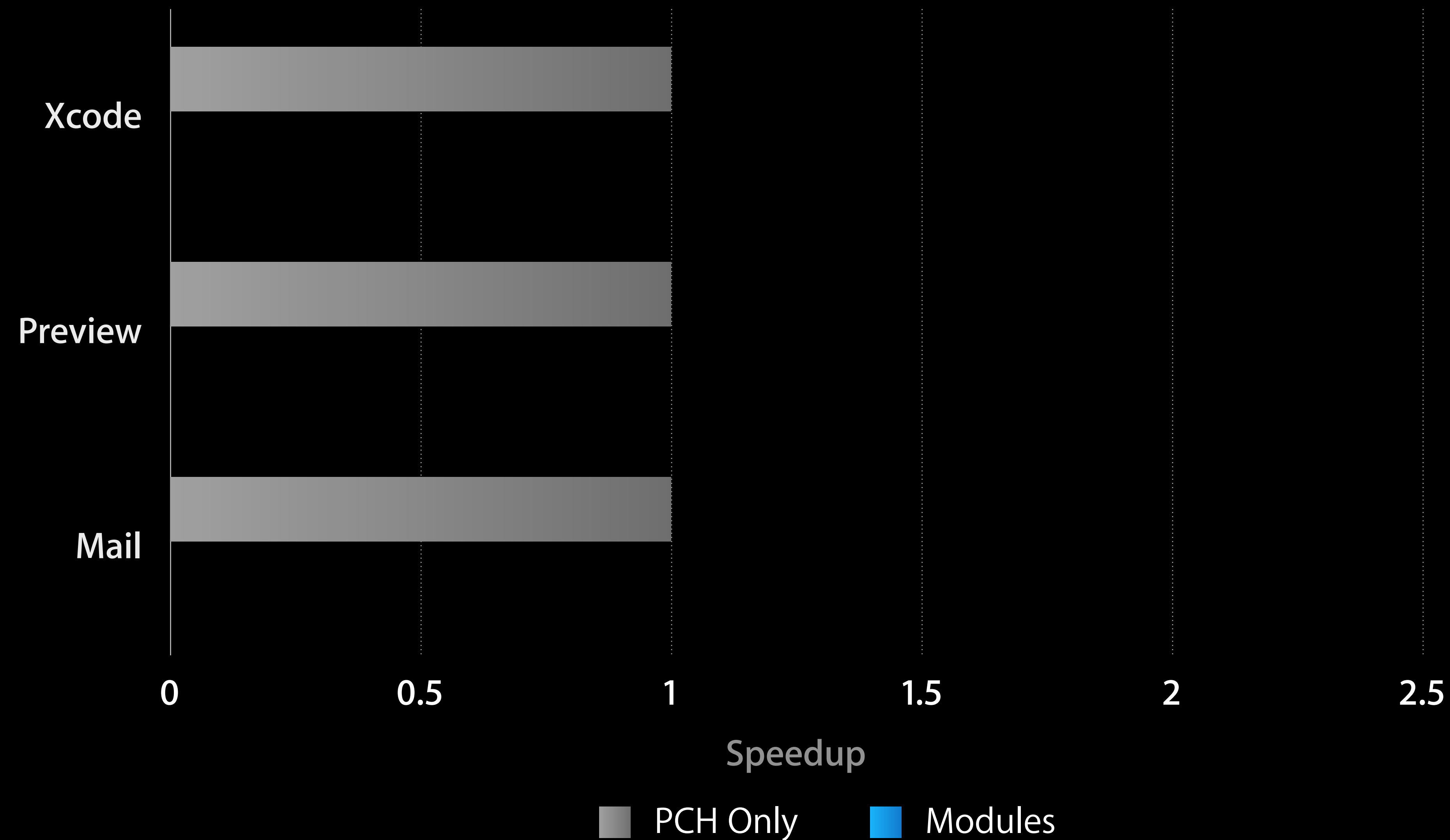
Build Time Improvements



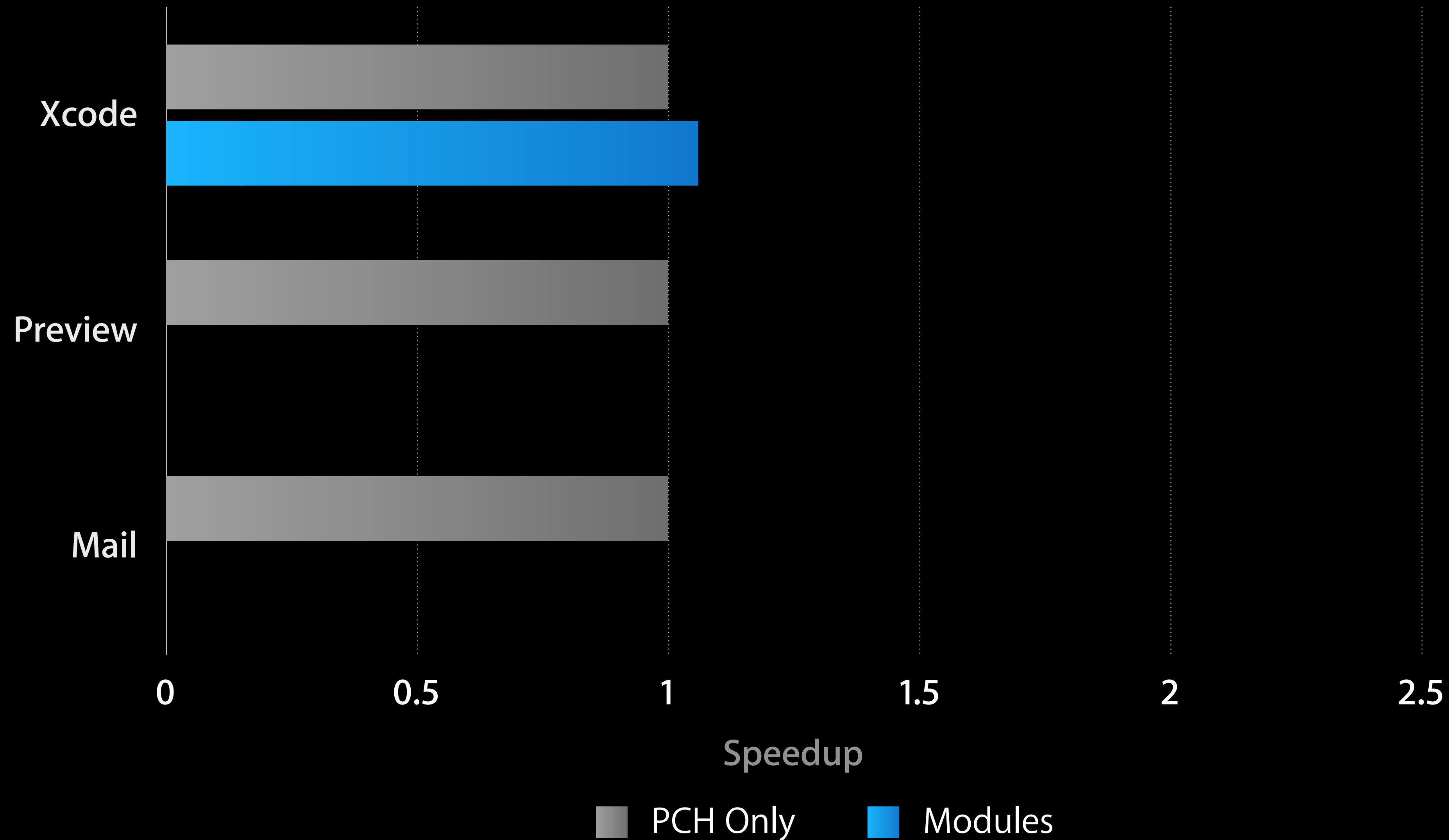
Build Time Improvements



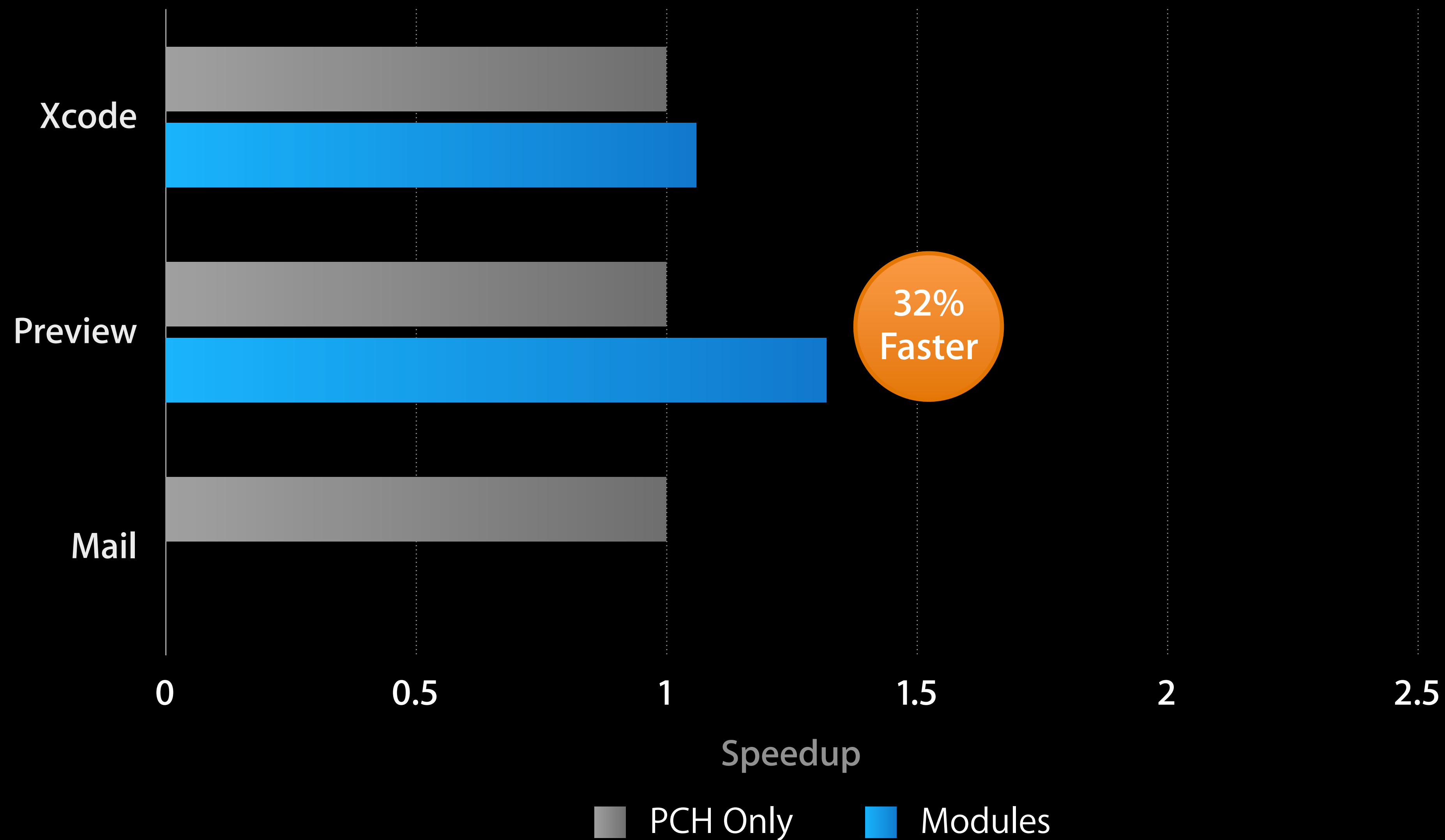
Indexing Time Improvements



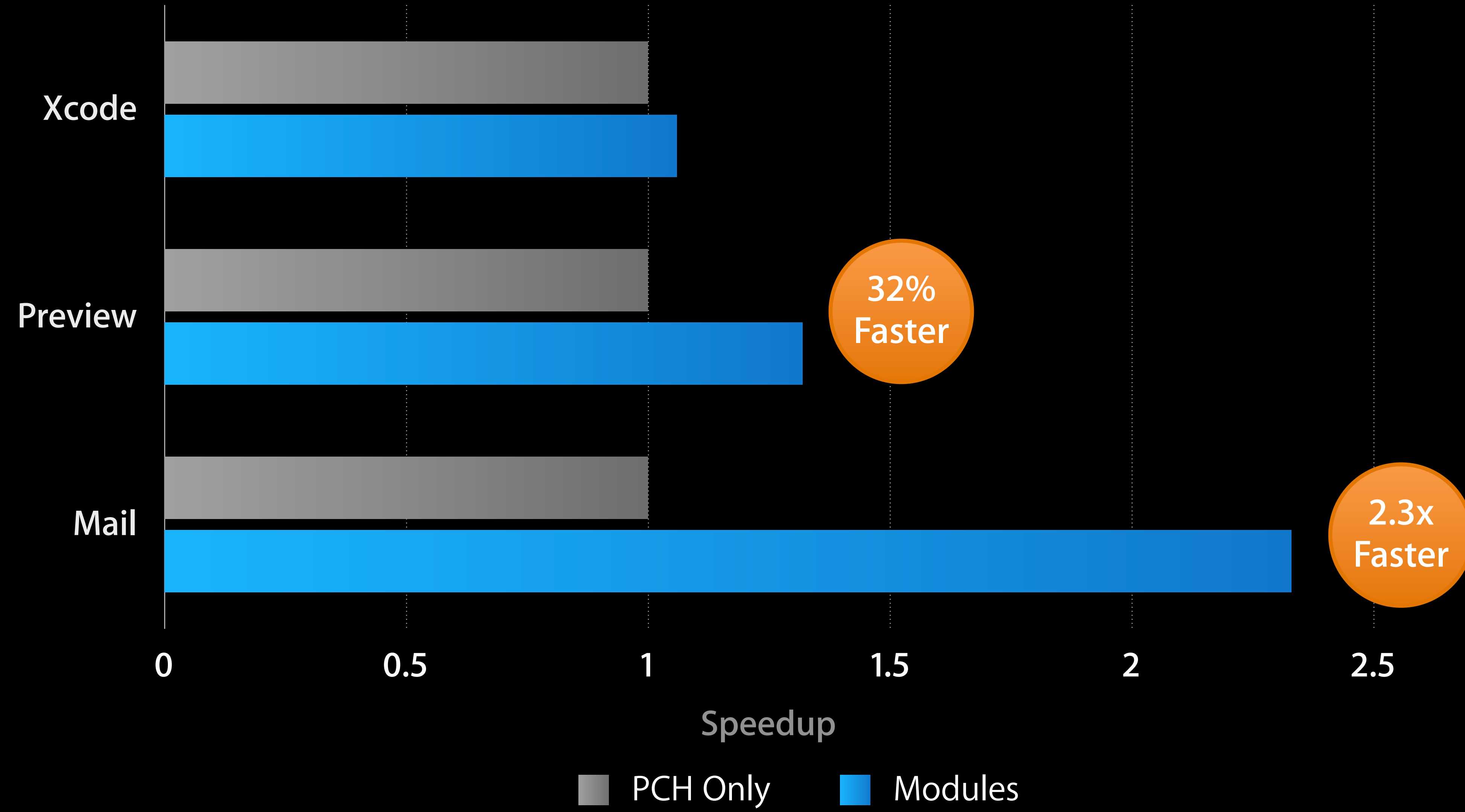
Indexing Time Improvements



Indexing Time Improvements

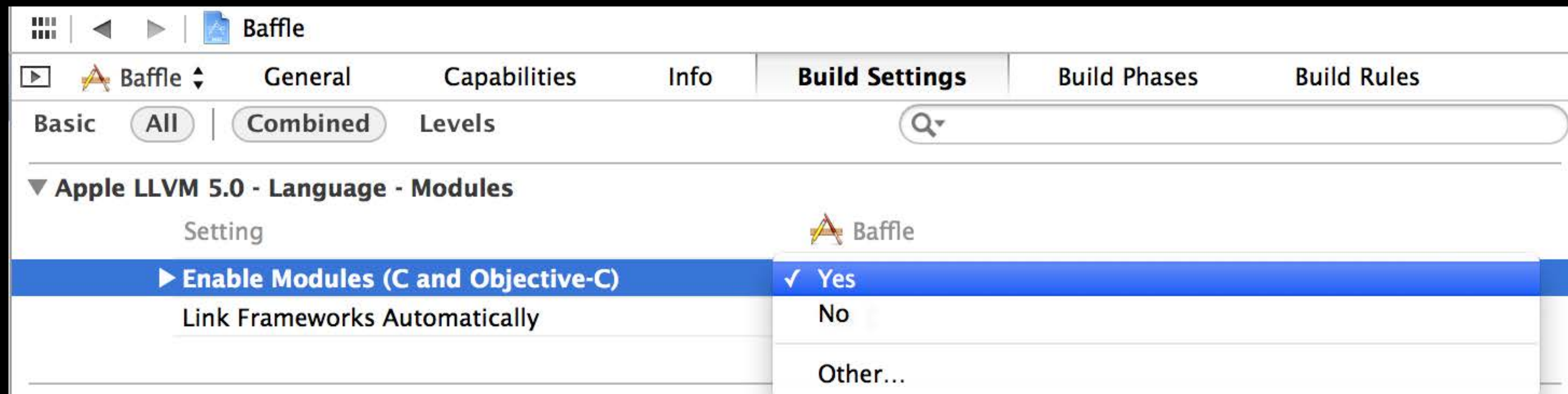


Indexing Time Improvements



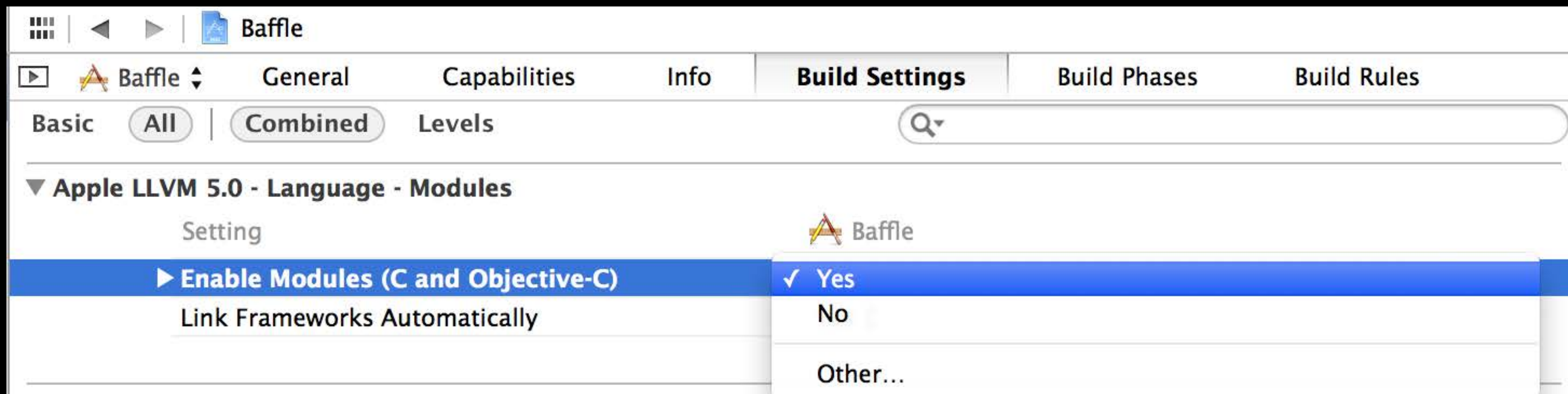
Enabling Modules

- Enabled by default in new projects



Enabling Modules

- Enabled by default in new projects

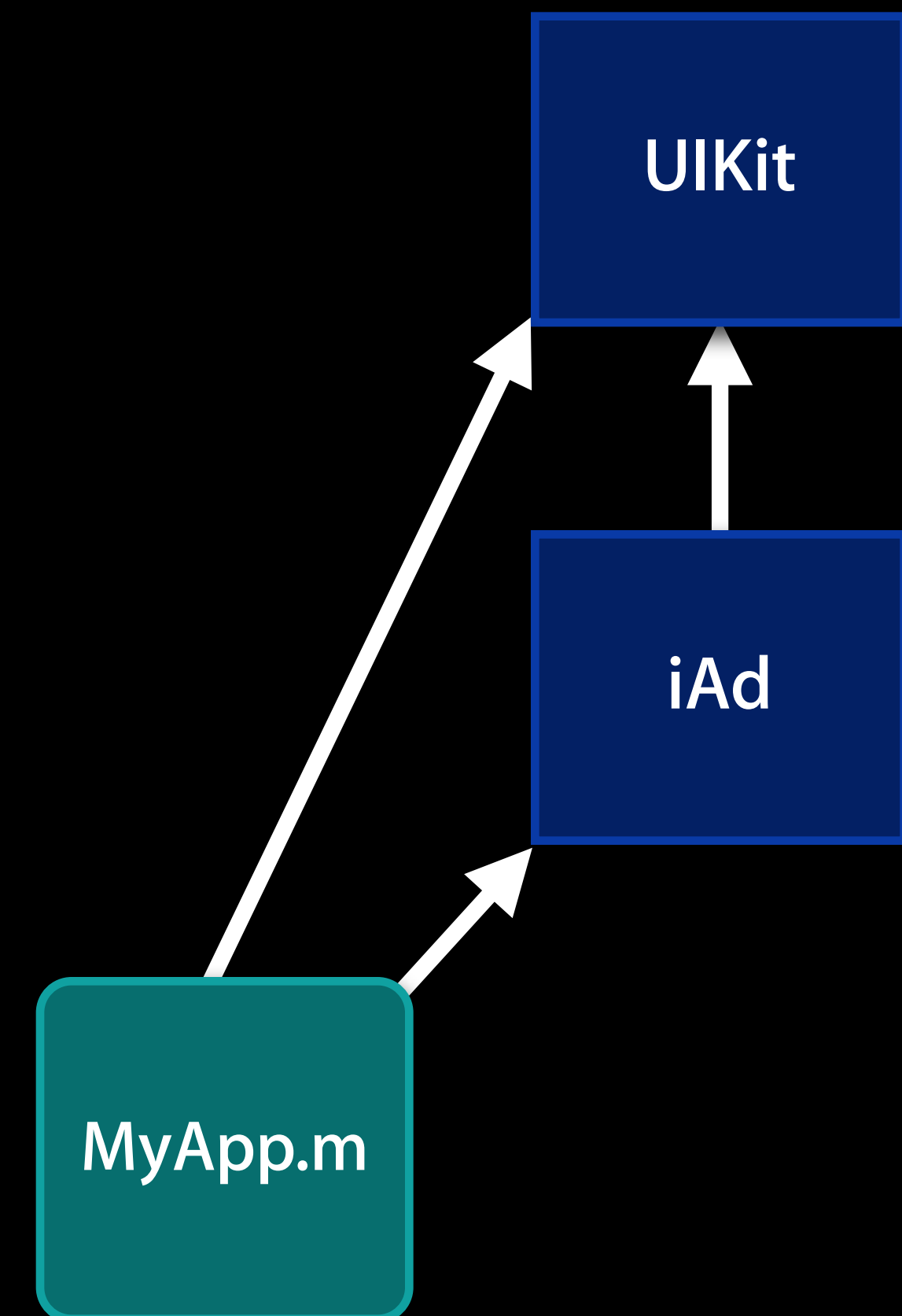


- Caveats:
 - Requires iOS 7 / OS X 10.9 SDK
 - Modules implicitly disabled for C++ sources
 - Modules not available for user frameworks

Modules Summary



- Simplify the use of frameworks
 - Semantic import rather than textual inclusion
 - Eliminate separate “link with libraries” step
- Improve performance of source tools
- No source changes required



Advances in Objective-C

Dave Zarzycki
Compiler Runtime Manager

Advances in Objective-C

Advances in Objective-C

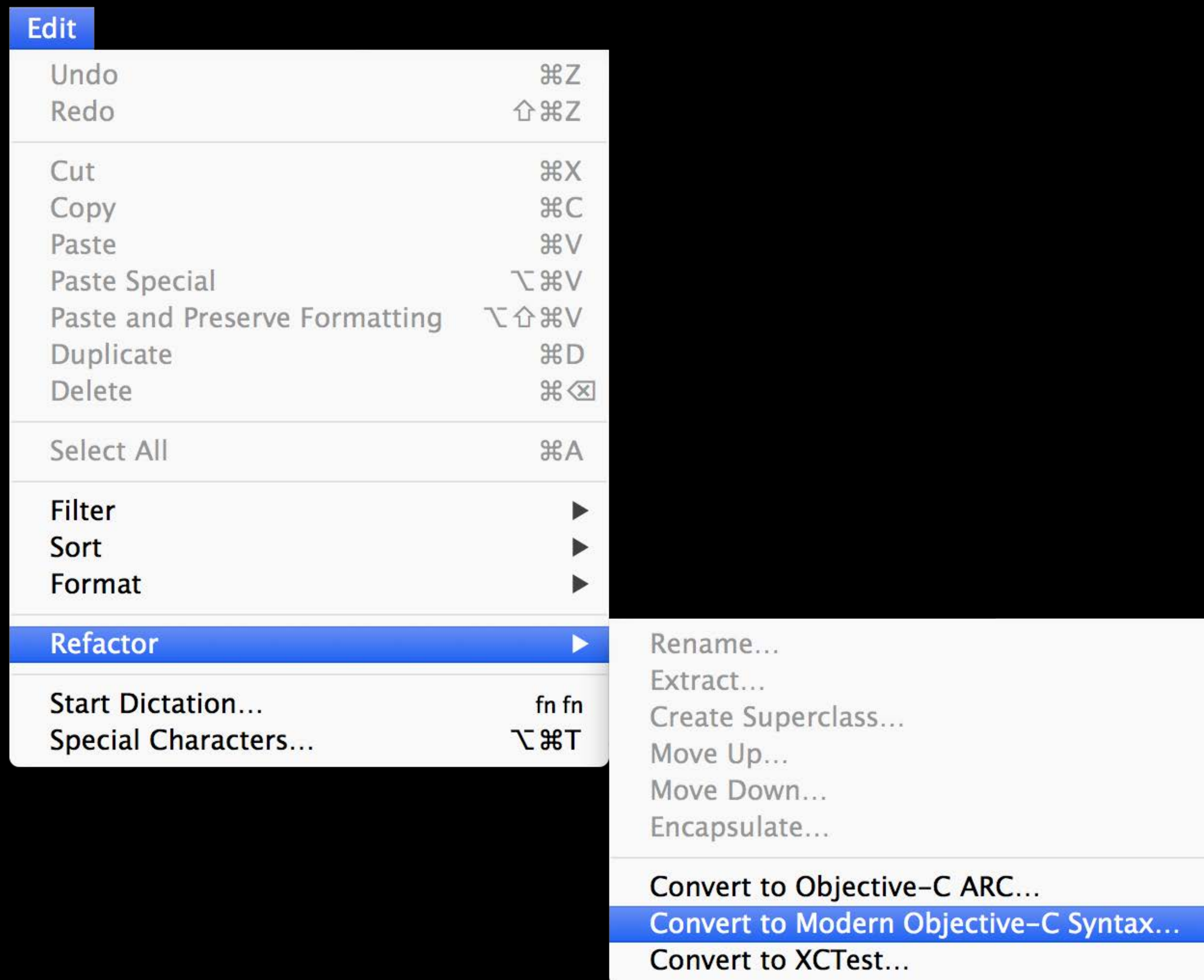
- Better productivity
 - Tools support for modernization
 - SDK improvements
 - Block return-type safety
 - The runtime and you

Advances in Objective-C

- Better productivity
 - Tools support for modernization
 - SDK improvements
 - Block return-type safety
 - The runtime and you
- ARC
 - Updates
 - Improvements

Tools Support for Modernization

Easiest change you can make today!



Reducing Boilerplate

- Object literals
- Container literals
- Subscripting
- Covered in depth during WWDC 2012
 - See talk 405—Modern Objective-C

Literals Before Modern Syntax

```
-(NSDictionary *)example {  
    return [NSDictionary dictionaryWithObjectsAndKeys:  
        @"Willie", @"PreferredName",  
        @"The Lion", @"NickName",  
        @"Smith", @"LastName",  
        @"William", @"FirstName",  
        [NSArray arrayWithObjects: @"Henry", @"Joseph", @"Bonaparte",  
            @"Bertholoff", nil], @"MiddleNames",  
        [NSNumber numberWithInt: 79], @"Age",  
        [NSNumber numberWithInt: 1893], @"BirthYear",  
        [NSNumber numberWithInt: 1973], @"DeathYear",  
        [NSNumber numberWithBool: YES], @"Male",  
        nil];  
}
```

Literals Before Modern Syntax

```
-(NSDictionary *)example {  
    return [NSDictionary dictionaryWithObjectsAndKeys:  
        @"Willie", @"PreferredName",  
        @"The Lion", @"NickName",  
        @"Smith", @"LastName",  
        @"William", @"FirstName",  
        [NSArray arrayWithObjects: @"Henry", @"Joseph", @"Bonaparte",  
            @"Bertholoff", nil], @"MiddleNames",  
        [NSNumber numberWithInt: 79], @"Age",  
        [NSNumber numberWithInt: 1893], @"BirthYear",  
        [NSNumber numberWithInt: 1973], @"DeathYear",  
        [NSNumber numberWithBool: YES], @"Male",  
        nil];  
}
```

Literals After Modern Syntax

```
-(NSDictionary *)example {  
    return @{  
        @"PreferredName": @"Willie",  
        @"NickName":      @"The Lion",  
        @"LastName":      @"Smith",  
        @"FirstName":      @"William",  
  
        @"MiddleNames":    @[ @"Henry", @"Joseph", @"Bonaparte", @"Bertholoff" ],  
        @"Age":             @79,  
        @"BirthYear":       @1893,  
        @"DeathYear":       @1973,  
        @"Male":            @YES  
    };  
}
```

Containers Before Modern Syntax

```
-(NSString *)swap1:(NSString *)arg {  
    NSString *tmp = [_dict objectForKey: @"key"];  
    [_dict setObject: arg forKey: @"key"];  
    return tmp;  
}  
  
-(NSString *)swap2:(NSString *)arg {  
    NSString *tmp = [_array objectAtIndex: 0];  
    [_array replaceObjectAtIndex: 0 withObject: tmp];  
    return tmp;  
}
```


Containers Before Modern Syntax

```
-(NSString *)swap1:(NSString *)arg {  
    NSString *tmp = [_dict objectForKey: @"key"];  
    [_dict setObject: arg forKey: @"key"];  
    return tmp;  
}  
  
-(NSString *)swap2:(NSString *)arg {  
    NSString *tmp = [_array objectAtIndex: 0];  
    [_array replaceObjectAtIndex: 0 withObject: tmp];  
    return tmp;  
}
```

Containers After Modern Syntax

```
-(NSString *)swap1:(NSString *)arg {  
    NSString *tmp = _dict[@"key"];  
    _dict[@"key"] = arg;  
    return tmp;  
}
```

```
-(NSString *)swap2:(NSString *)arg {  
    NSString *tmp = _array[0];  
    _array[0] = tmp;  
    return tmp;  
}
```

More to Modern Syntax

More to Modern Syntax

- Boxed expressions via @()

More to Modern Syntax

- Boxed expressions via `@()`
- Full interaction with C types

More to Modern Syntax

- Boxed expressions via `@()`
- Full interaction with C types
- How to implement subscripting for your objects

More to Modern Syntax

- Boxed expressions via `@()`
- Full interaction with C types
- How to implement subscripting for your objects
- Covered in depth during WWDC 2012

More to Modern Syntax

- Boxed expressions via `@()`
- Full interaction with C types
- How to implement subscripting for your objects
- Covered in depth during WWDC 2012
 - See 405—Modern Objective-C

SDK Improvements



SDK Improvements

- Leveraging the improving compiler
 - Better correctness
 - Better safety
 - Better compile-time error detection



SDK Improvements

- Leveraging the improving compiler
 - Better correctness
 - Better safety
 - Better compile-time error detection
- New features and you
 - The “instancetype” keyword
 - Explicitly-typed enums



Return Type Correctness

Return Type Correctness

```
-(NSDictionary *)exampleFactoryUsage {  
    NSDictionary *var = [NSArray array];  
    return var;  
}
```

Return Type Correctness

```
-(NSDictionary *)exampleFactoryUsage {  
    NSDictionary *var = [NSArray array];  
    return var;  
}
```

Return Type Correctness

```
-(NSDictionary *)exampleFactoryUsage {  
    NSDictionary *var = [NSArray array];  
    return var;  
}
```

- Copy-and-paste errors are easy

Return Type Correctness

```
-(NSDictionary *)exampleFactoryUsage {  
    NSDictionary *var = [NSArray array];  
    return var;  
}
```

- Copy-and-paste errors are easy
- Refactoring errors are easy

Return Type Correctness

```
-(NSDictionary *)exampleFactoryUsage {  
    NSDictionary *var = [NSArray array];  
    return var;  
}
```

warning: incompatible pointer types initializing
'NSDictionary *' with an expression of type 'NSArray
*' [-Wincompatible-pointer-types]

```
    NSDictionary *var = [NSArray array];
```

^

~~~~~

# How Does the Compiler Know?

# How Does the Compiler Know?

```
+(id)array;
```

# How Does the Compiler Know?

```
+(id)array;
```

- Implicitly converts to any object type

# How Does the Compiler Know?

```
+(instancetype)array;
```

# How Does the Compiler Know?

```
+(instancetype)array;
```

- A contextual keyword

# How Does the Compiler Know?

```
+(instancetype)array;
```

- A contextual keyword
- Only for return types

# How Does the Compiler Know?

```
+(instancetype)array;
```

- A contextual keyword
- Only for return types
- Subclasses do not need to redeclare “instancetype” methods



# How Does the Compiler Know?

```
+(instancetype)array;
```

- A contextual keyword
- Only for return types
- Subclasses do not need to redeclare “instancetype” methods
- The compiler contextually matches the return type to the receiver

# Subclasses and “instancetype”

Implicitly does what you want

# Subclasses and “instancetype”

Implicitly does what you want

```
@interface Foobar : NSArray
@end
// ...
NSDictionary *var = [Foobar array];
```

# Subclasses and “instancetype”

Implicitly does what you want

```
@interface Foobar : NSArray
@end
// ...
NSDictionary *var = [Foobar array];
```

**warning:** incompatible pointer types initializing  
'NSDictionary \*' with an expression of type 'Foobar \*' [-  
Wincompatible-pointer-types]

```
    NSDictionary *var = [Foobar array];
```

^

~~~~~

Subclasses and “instancetype”

Implicitly does what you want

```
@interface Foobar : NSArray
@end
// ...
NSDictionary *var = [Foobar array];
```

warning: incompatible pointer types initializing
'NSDictionary *' with an expression of type '**Foobar ***' [-
Wincompatible-pointer-types]

```
    NSDictionary *var = [Foobar array];
```

^

~~~~~

# Explicitly-Typed Enums

# Explicitly-Typed Enums

```
NSURLSessionStatus status = NSURLSessionTaskStateRunning;
```

# Explicitly-Typed Enums

```
NSURLSessionHandleStatus status = NSURLSessionTaskStateRunning;
```



# Explicitly-Typed Enums

```
NSURLSessionStatus status = NSURLSessionTaskStateRunning;
```

- Copy-and-paste errors are easy

# Explicitly-Typed Enums

```
NSURLSessionStatus status = NSURLSessionTaskStateRunning;
```

- Copy-and-paste errors are easy
- Refactoring errors are easy

# Explicitly-Typed Enums

```
NSURLSessionStatus status = NSURLSessionTaskStateRunning;
```

- Copy-and-paste errors are easy
- Refactoring errors are easy
- Enums are global integers

# Explicitly-Typed Enums

```
NSURLSessionHandleStatus status = NSURLSessionTaskStateRunning;
```

**warning:** implicit conversion from enumeration type 'enum NSURLSessionTaskState' to different enumeration type 'NSURLSessionHandleStatus' (aka 'enum NSURLHandleStatus')  
[-Wenum-conversion]

```
NSURLSessionHandleStatus status = NSURLSessionTaskStateRunning;
```



# How Does the Compiler Know?

```
enum { ABC, JKL, XYZ };  
typedef NSUInteger MyEnum;
```

# How Does the Compiler Know?

```
enum { ABC, JKL, XYZ };  
typedef NSUInteger MyEnum;
```

- Traditional C enums are implicitly “int”

# How Does the Compiler Know?

```
enum MyEnum : NSUInteger { ABC, JKL, XYZ };  
typedef MyEnum MyEnum;
```

- Traditional C enums are implicitly “int”
- Enums now support a fixed underlying type
- Covered in depth during WWDC 2012
  - See talk 405—Modern Objective-C

# Convenient Foundation Macros



# Convenient Foundation Macros

```
typedef NS_ENUM(NSUInteger, MyEnum) { ABC, JKL, XYZ };  
  
typedef NS_OPTIONS(NSUInteger, MyOptions) {  
    kFaster    = (1 << 3),  
    kBetter    = (1 << 4),  
    kAwesome   = (1 << 5)  
};
```

# Code Completion Before NS\_ENUM()

# Code Completion Before NS\_ENUM()

```
enum { ABC, JKL, XYZ };  
typedef NSUInteger MyEnum;
```

```
- (void)takeMyEnum:(MyEnum)e { }
```

```
- (void)enumExample {  
    [self takeMyEnum:xpc_array_get_count(xpc_object_t xarray)]
```

```
f xpc_connection_t xpc_array_create_connection(xpc_object_t xarray, size_t index)  
f int xpc_array_dup_fd(xpc_object_t xarray, size_t index)  
f bool xpc_array_get_bool(xpc_object_t xarray, size_t index)  
f size_t xpc_array_get_count(xpc_object_t xarray)  
f const void * xpc_array_get_data(xpc_object_t xarray, size_t index, size_t *length)  
f int64_t xpc_array_get_date(xpc_object_t xarray, size_t index)  
f double xpc_array_get_double(xpc_object_t xarray, size_t index)  
f int64_t xpc_array_get_int64(xpc_object_t xarray, size_t index)
```

Returns the count of values currently in the array. [More...](#)

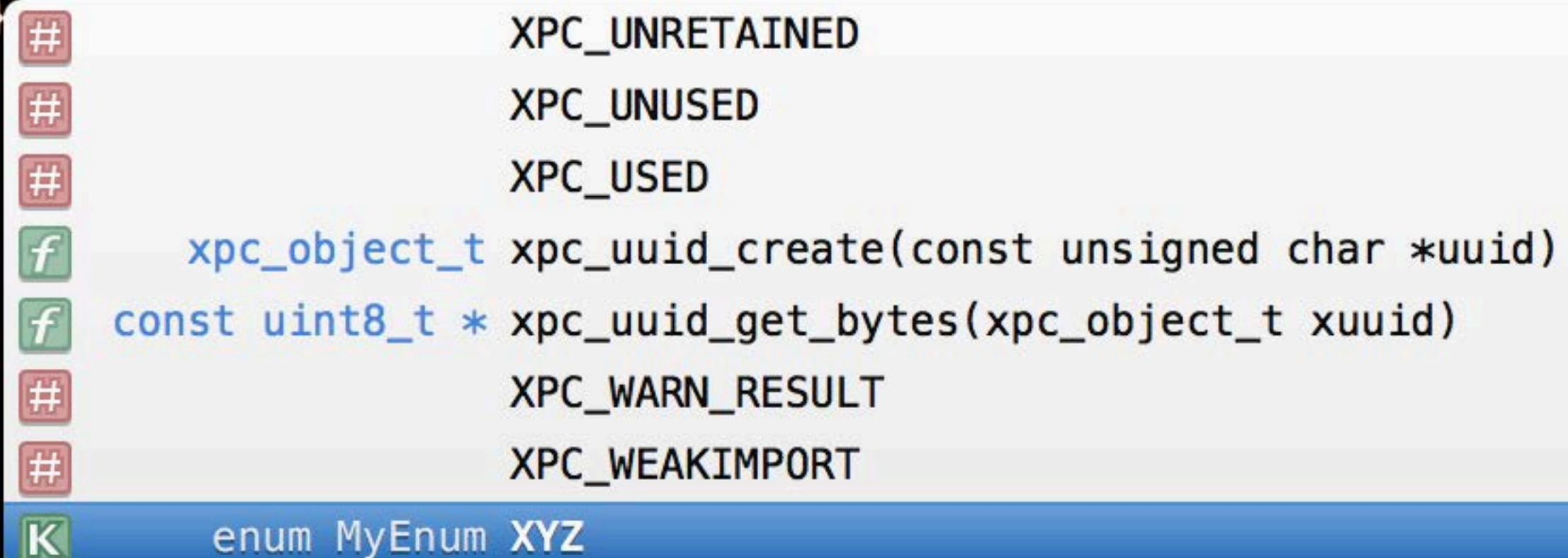


# Code Completion After NS\_ENUM()

```
typedef NS_ENUM(NSUInteger, MyEnum) { ABC, JKL, XYZ };
```

```
- (void)takeMyEnum:(MyEnum)e { }
```

```
- (void)enumExample {  
    [self takeMyEnum:XYZ]  
}
```

A screenshot of an Xcode code completion menu. The menu is a light gray rectangle with a blue bar at the bottom. It contains a list of suggestions, each preceded by a small icon in a red square. The suggestions are: XPC\_UNRETAINED (hash icon), XPC\_UNUSED (hash icon), XPC\_USED (hash icon), xpc\_object\_t xpc\_uuid\_create(const unsigned char \*uuid) (function icon), const uint8\_t \* xpc\_uuid\_get\_bytes(xpc\_object\_t xuuid) (function icon), XPC\_WARN\_RESULT (hash icon), XPC\_WEAKIMPORT (hash icon), and enum MyEnum XYZ (key icon).

```
# XPC_UNRETAINED  
# XPC_UNUSED  
# XPC_USED  
f xpc_object_t xpc_uuid_create(const unsigned char *uuid)  
f const uint8_t * xpc_uuid_get_bytes(xpc_object_t xuuid)  
# XPC_WARN_RESULT  
# XPC_WEAKIMPORT  
K enum MyEnum XYZ
```

# Return-type Inference

```
myNSArray = [myNSArray sortedArrayUsingComparator: ^(id lhs, id rhs) {  
    if (...) {  
        return NSOrderedAscending;  
    } else {  
        return NSOrderedDescending;  
    }  
}]
```

# Return-type Inference

```
myNSArray = [myNSArray sortedArrayUsingComparator: ^(id lhs, id rhs) {  
    if (...) {  
        return NSOrderedAscending;  
    } else {  
        return NSOrderedDescending;  
    }  
}]
```

**error:** incompatible block pointer types sending 'int (^)(id, id)' to parameter of type 'NSComparator' (aka 'NSComparisonResult (^)(id, id)')

```
myNSArray = [myNSArray sortedArrayUsingComparator: ^(id lhs, id rhs) {
```

^

# Return-type Inference

```
myNSArray = [myNSArray sortedArrayUsingComparator: ^(id lhs, id rhs) {  
    if (...) {  
        return (NSComparisonResult)NSOrderedAscending;  
    } else {  
        return (NSComparisonResult)NSOrderedDescending;  
    }  
}]
```

**error:** incompatible block pointer types sending 'int (^)(id, id)' to parameter of type 'NSComparator' (aka 'NSComparisonResult (^)(id, id)')

```
myNSArray = [myNSArray sortedArrayUsingComparator: ^(id lhs, id rhs) {  
    ^
```

# Return-type Inference

```
myNSArray = [myNSArray sortedArrayUsingComparator: ^(id lhs, id rhs) {  
    if (...) {  
        return (NSComparisonResult)NSOrderedAscending;  
    } else {  
        return (NSComparisonResult)NSOrderedDescending;  
    }  
}]
```



# Return-type Inference

```
myNSArray = [myNSArray sortedArrayUsingComparator: ^(id lhs, id rhs) {  
    if (...) {  
        return (NSComparisonResult)NSOrderedAscending;  
    } else {  
        return (NSComparisonResult)NSOrderedDescending;  
    }  
}]
```

- Implicitly typed enums can require more casting
- NS\_ENUM() helps you avoid casting

# Return-type Inference

```
myNSArray = [myNSArray sortedArrayUsingComparator: ^(id lhs, id rhs) {  
    if (...) {  
        return NSOrderedAscending;  
    } else {  
        return NSOrderedDescending;  
    }  
}]
```

- Implicitly typed enums can require more casting
- `NS_ENUM()` helps you avoid casting

# Return-type Inference

```
-(void)returnInference:(BOOL)arg {  
    someAPI(^{  
        if (arg) return NSURLHandleLoadSucceeded;  
        else return NSURLSessionTaskStateRunning;  
    });  
}
```

# Return-type Inference

```
-(void)returnInference:(BOOL)arg {  
    someAPI(^{  
        if (arg) return NSURLHandleLoadSucceeded;  
        else return NSURLSessionTaskStateRunning;  
    });  
}
```

- Implicit enums create silent bugs
- NS\_ENUM() helps the compiler produce an error

# Return-type Inference

```
-(void)returnInference:(BOOL)arg {  
    someAPI(^{  
        if (arg) return NSURLHandleLoadSucceeded;  
        else return NSURLSessionTaskStateRunning;  
    });  
}
```

**error:** return type 'NSURLSessionTaskState' must match previous return type 'NSURLHandleStatus' when block literal has unspecified explicit return type

else NSURLSessionTaskStateRunning;

^

# The Objective-C Runtime

The core of the language

# The Objective-C Runtime

## The core of the language

- Enables dynamic behavior
- Method dispatch
- Object introspection
- Object proxies
- Dynamic class construction and replacement

# The Runtime Enables Innovation



# The Runtime Enables Innovation

- Many features have been added over the years

# The Runtime Enables Innovation

- Many features have been added over the years
- The heart of these features are in the runtime

# The Runtime Enables Innovation

- Many features have been added over the years
- The heart of these features are in the runtime
- Examples:
  - Key-Value observing
  - Associated objects
  - @synchronized
  - Weak references
  - Tagged pointers
  - etc.

# Tagged Pointers

Example innovation

# Tagged Pointers

## Example innovation

- Added to 64-bit Cocoa
  - For small value-like objects
  - NSNumber, NSDate, etc.

# Tagged Pointers

## Example innovation

- Added to 64-bit Cocoa
  - For small value-like objects
  - NSNumber, NSDate, etc.
- Stores object in the pointer itself
  - No malloc/free overhead

# Tagged Pointers

## Example innovation

- Added to 64-bit Cocoa
  - For small value-like objects
  - NSNumber, NSDate, etc.
- Stores object in the pointer itself
  - No malloc/free overhead
- Performance
  - Three times more space efficient!
  - 106 times faster to allocate/destroy!

# How Tagged Pointers Work

Optimizing bits



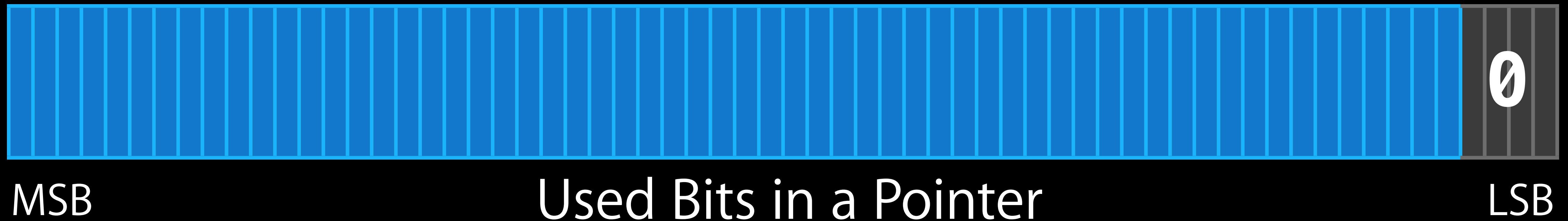
MSB

LSB



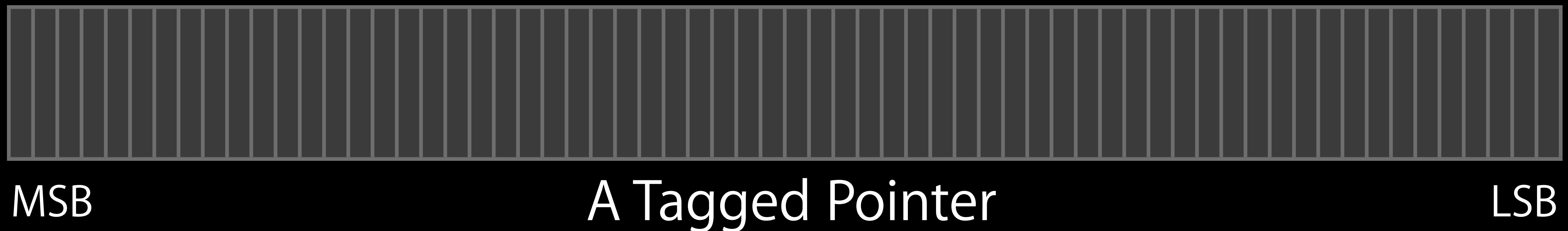
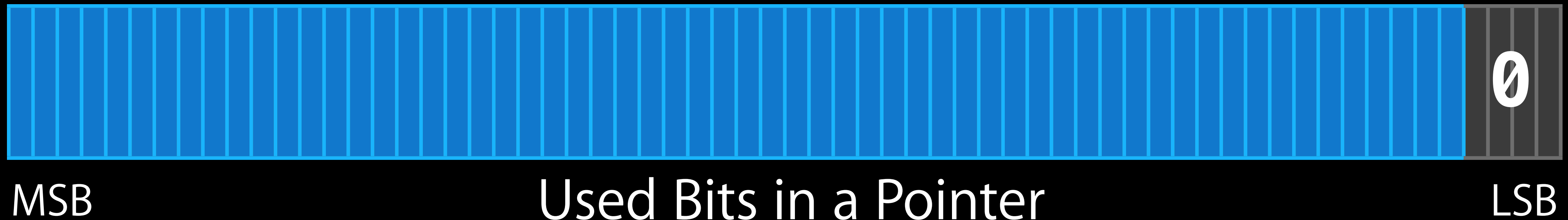
# How Tagged Pointers Work

Optimizing bits



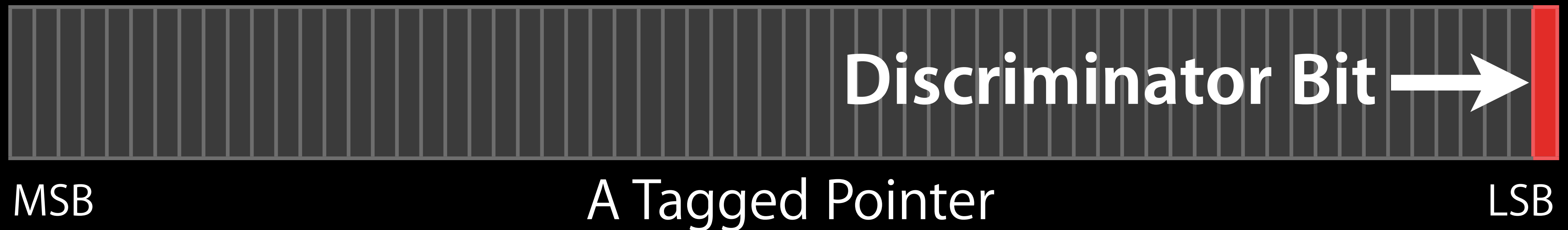
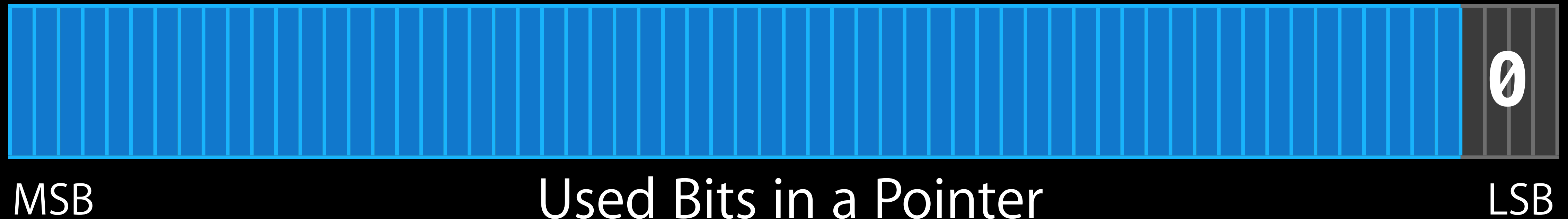
# How Tagged Pointers Work

Optimizing bits



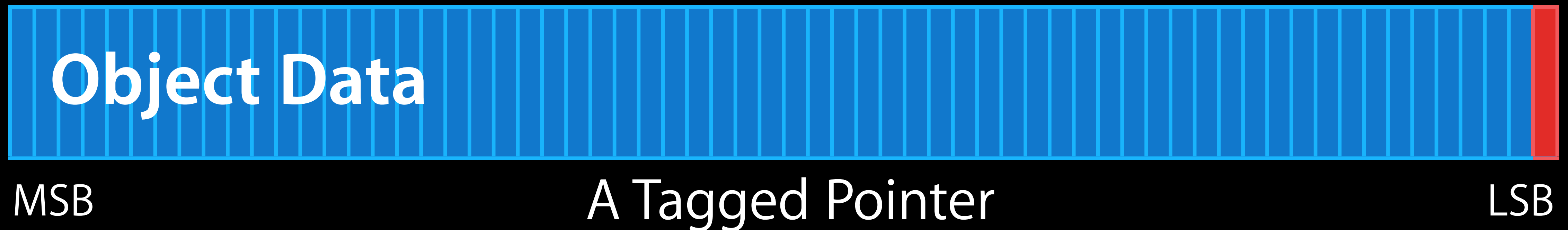
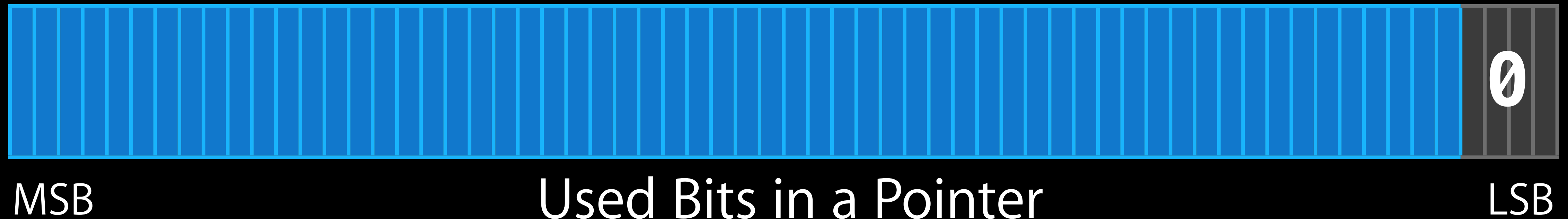
# How Tagged Pointers Work

Optimizing bits



# How Tagged Pointers Work

Optimizing bits



# Tagged Pointers and You

An implementation detail

# Tagged Pointers and You

## An implementation detail

- The runtime data is almost all private
  - The remaining public data structures are becoming private

# Tagged Pointers and You

## An implementation detail

- The runtime data is almost all private
  - The remaining public data structures are becoming private
- Most apps are well behaved
  - Use API to introspect or update
  - This lets us innovate considerably!

# Tagged Pointers and You

## An implementation detail

- The runtime data is almost all private
  - The remaining public data structures are becoming private
- Most apps are well behaved
  - Use API to introspect or update
  - This lets us innovate considerably!
- New warnings
  - Tagged pointers
  - Raw 'isa' access



# New Tagged Pointer and Raw ISA Warnings

```
-(BOOL)exampleTagUsage:(NSObject *)arg {  
    if (((long)arg & 1) == 0) return arg->isa == cachedValue;  
    else return [arg isKindOfClass: cachedValue];  
}
```

# New Tagged Pointer and Raw ISA Warnings

```
-(BOOL)exampleTagUsage:(NSObject *)arg {  
    if (((long)arg & 1) == 0) return arg->isa == cachedValue;  
    else return [arg isKindOfClass: cachedValue];  
}
```

**warning:** bitmasking for introspection of Objective-C object pointers is strongly discouraged [-Wdeprecated-objc-pointer-introspection]

```
    if (((long)arg & 1) == 0) return arg->isa == cachedValue;
```

~~~~~ ^

New Tagged Pointer and Raw ISA Warnings

```
-(BOOL)exampleTagUsage:(NSObject *)arg {  
    if (((long)arg & 1) == 0) return arg->isa == cachedValue;  
    else return [arg isKindOfClass: cachedValue];  
}
```

warning: bitmasking for introspection of Objective-C object pointers is strongly discouraged [-Wdeprecated-objc-pointer-introspection]

```
    if (((long)arg & 1) == 0) return arg->isa == cachedValue;
```

~~~~~ ^

**error:** direct access to Objective-C's isa is deprecated in favor of object\_getClass() [-Werror,-Wdeprecated-objc-isa-usage]

```
    if (((long)arg & 1) == 0) return arg->isa == cachedValue;
```

^

# New Tagged Pointer and Raw ISA Warnings

```
-(BOOL)exampleTagUsage:(NSObject *)arg {  
    return [arg isKindOfClass: cachedValue];  
}
```

- We want to unlock the next level of innovation
- Please use `-isKindOfClass:` or `object_getClass()`
- Failure to do so may break your code in the future!



# Garbage Collection

# Garbage Collection

- Only available on the Mac

# Garbage Collection

- Only available on the Mac
- Replaced by ARC

# Garbage Collection

- Only available on the Mac
- Replaced by ARC
- Deprecated with OSX 10.8



# Garbage Collection

- Only available on the Mac
- Replaced by ARC
- Deprecated with OSX 10.8
- Not supported by new frameworks
  - AVKit, Accounts, GameController, GameKit, MapKit, Social, SpriteKit, etc.

# Garbage Collection

- Only available on the Mac
- Replaced by ARC
- Deprecated with OSX 10.8
- Not supported by new frameworks
  - AVKit, Accounts, GameController, GameKit, MapKit, Social, SpriteKit, etc.
- *Please* use the ARC migrator to transition off GC



# Automatic Reference Counting

Updates and improvements

# ARC Update

# ARC Update

- Cocoa is designed with reference counting semantics
  - Deterministic object destruction order is important!
  - Great for debugging too

# ARC Update

- Cocoa is designed with reference counting semantics
  - Deterministic object destruction order is important!
  - Great for debugging too
- ARC helps you write great Cocoa code
  - Allows you to focus on what matters, your app

# ARC Update

- Cocoa is designed with reference counting semantics
  - Deterministic object destruction order is important!
  - Great for debugging too
- ARC helps you write great Cocoa code
  - Allows you to focus on what matters, your app
- Majority of new app store submissions use ARC

# ARC and Xcode 5.0



# ARC and Xcode 5.0

- Xcode now uses ARC
  - Was a large GC app

# ARC and Xcode 5.0

- Xcode now uses ARC
  - Was a large GC app
- Better developer experience
  - Determinism
  - Debugging
  - Performance

# ARC and Performance

# ARC and Performance

- Continuous improvement

# ARC and Performance

- Continuous improvement
- `__weak` references are about 2x faster on iOS 7.0 and OSX 10.9

# ARC and Performance

- Continuous improvement
- `__weak` references are about 2x faster on iOS 7.0 and OSX 10.9
- More predictable memory usage in debug builds

# ARC and Performance

- Continuous improvement
- `__weak` references are about 2x faster on iOS 7.0 and OSX 10.9
- More predictable memory usage in debug builds
- Lifetime of autoreleased objects is more like release builds

# ARC Migrator



# ARC Migrator

- The migrator does the “heavy lifting”
  - Removes retain/release/autorelease
  - Removes empty dealloc methods
  - Converts NSAutoreleasePool to @autoreleasepool

# ARC Migrator

- The migrator does the “heavy lifting”
  - Removes retain/release/autorelease
  - Removes empty dealloc methods
  - Converts NSAutoreleasePool to @autoreleasepool
- You do the rest
  - “id” in structs (rare)
  - Atypical uses of memory management API

# ARC Migrator

- The migrator does the “heavy lifting”
  - Removes retain/release/autorelease
  - Removes empty dealloc methods
  - Converts NSAutoreleasePool to @autoreleasepool
- You do the rest
  - “id” in structs (rare)
  - Atypical uses of memory management API
- Covered in depth during WWDC 2012

# ARC Migrator

Edit

|                               |       |
|-------------------------------|-------|
| Undo                          | ⌘Z    |
| Redo                          | ⇧⌘Z   |
| Cut                           | ⌘X    |
| Copy                          | ⌘C    |
| Paste                         | ⌘V    |
| Paste Special                 | ⇧⌘V   |
| Paste and Preserve Formatting | ⇧⇧⌘V  |
| Duplicate                     | ⌘D    |
| Delete                        | ⌘⌫    |
| Select All                    | ⌘A    |
| Filter                        | ▶     |
| Sort                          | ▶     |
| Format                        | ▶     |
| Refactor                      | ▶     |
| Start Dictation...            | fn fn |
| Special Characters...         | ⇧⌘T   |

|                                         |
|-----------------------------------------|
| Rename...                               |
| Extract...                              |
| Create Superclass...                    |
| Move Up...                              |
| Move Down...                            |
| Encapsulate...                          |
| Convert to Objective-C ARC...           |
| Convert to Modern Objective-C Syntax... |
| Convert to XCTest...                    |

# ARC and Your App

# ARC and Your App



- Switch to ARC by default
  - Can opt out specific files

# ARC and Your App



- Switch to ARC by default
  - Can opt out specific files
- The ARC migrator supports
  - Manual retain/release code
  - Garbage-collected code

# New Memory Management Warnings





# New Memory Management Warnings



- Help you reason about object lifetime

# New Memory Management Warnings



- Help you reason about object lifetime
- Implicit retain of 'self' within blocks

# New Memory Management Warnings



- Help you reason about object lifetime
- Implicit retain of 'self' within blocks
- Repeatedly using a `__weak` reference

# New Memory Management Warnings



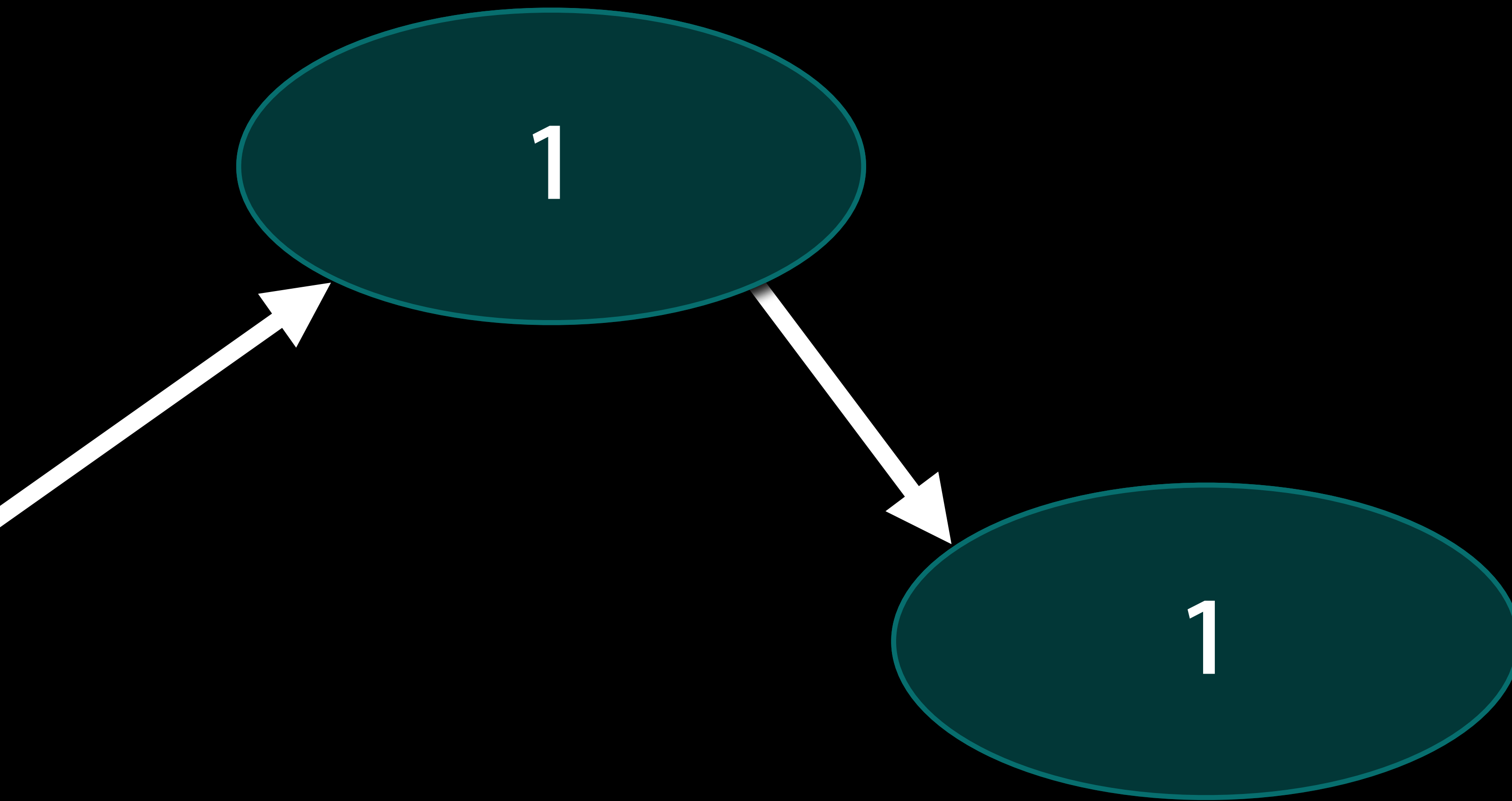
- Help you reason about object lifetime
- Implicit retain of 'self' within blocks
- Repeatedly using a `__weak` reference
- Sending messages to `__weak` pointers

# Understanding Retain Cycles

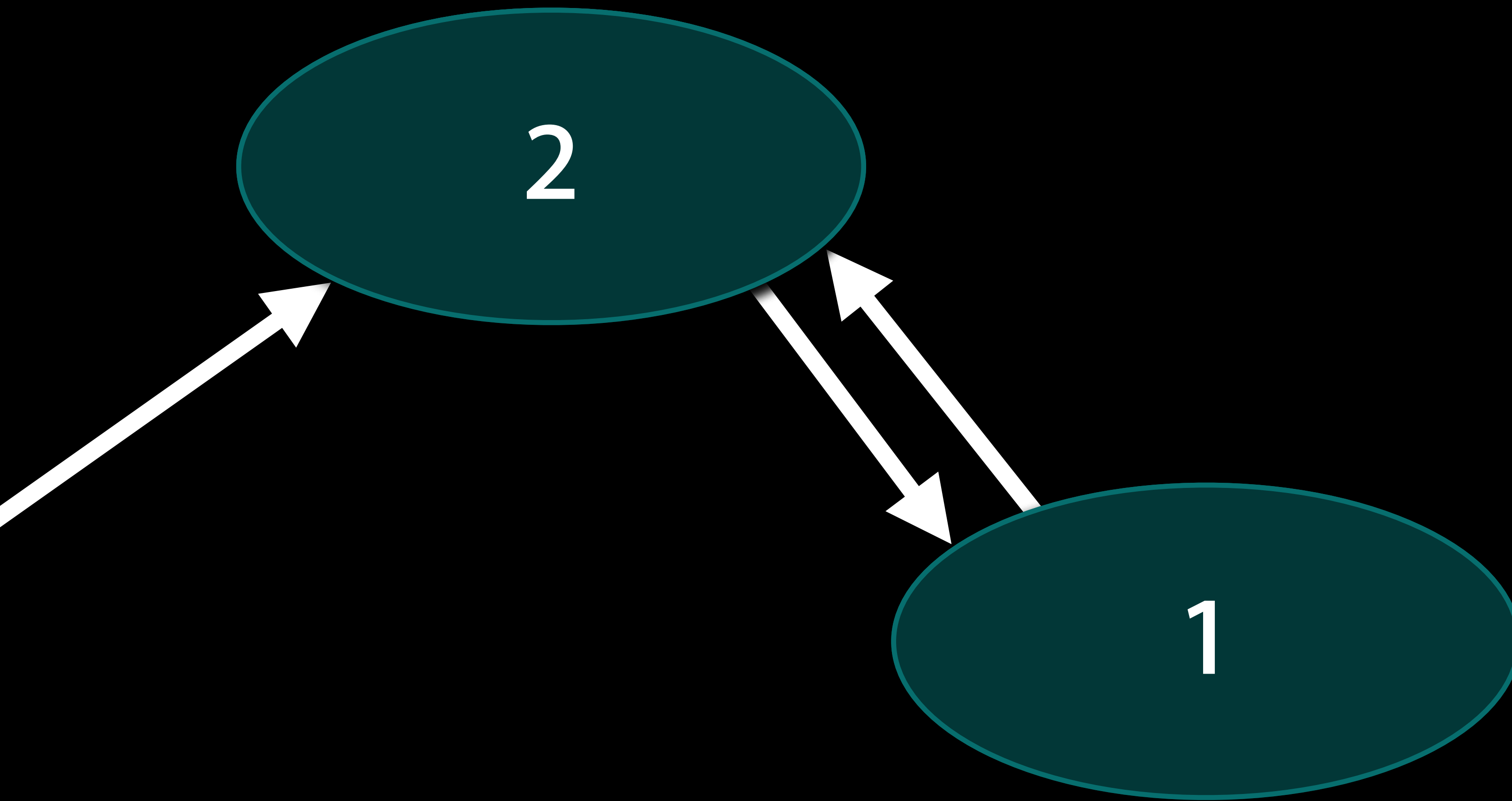
# Understanding Retain Cycles



# Understanding Retain Cycles

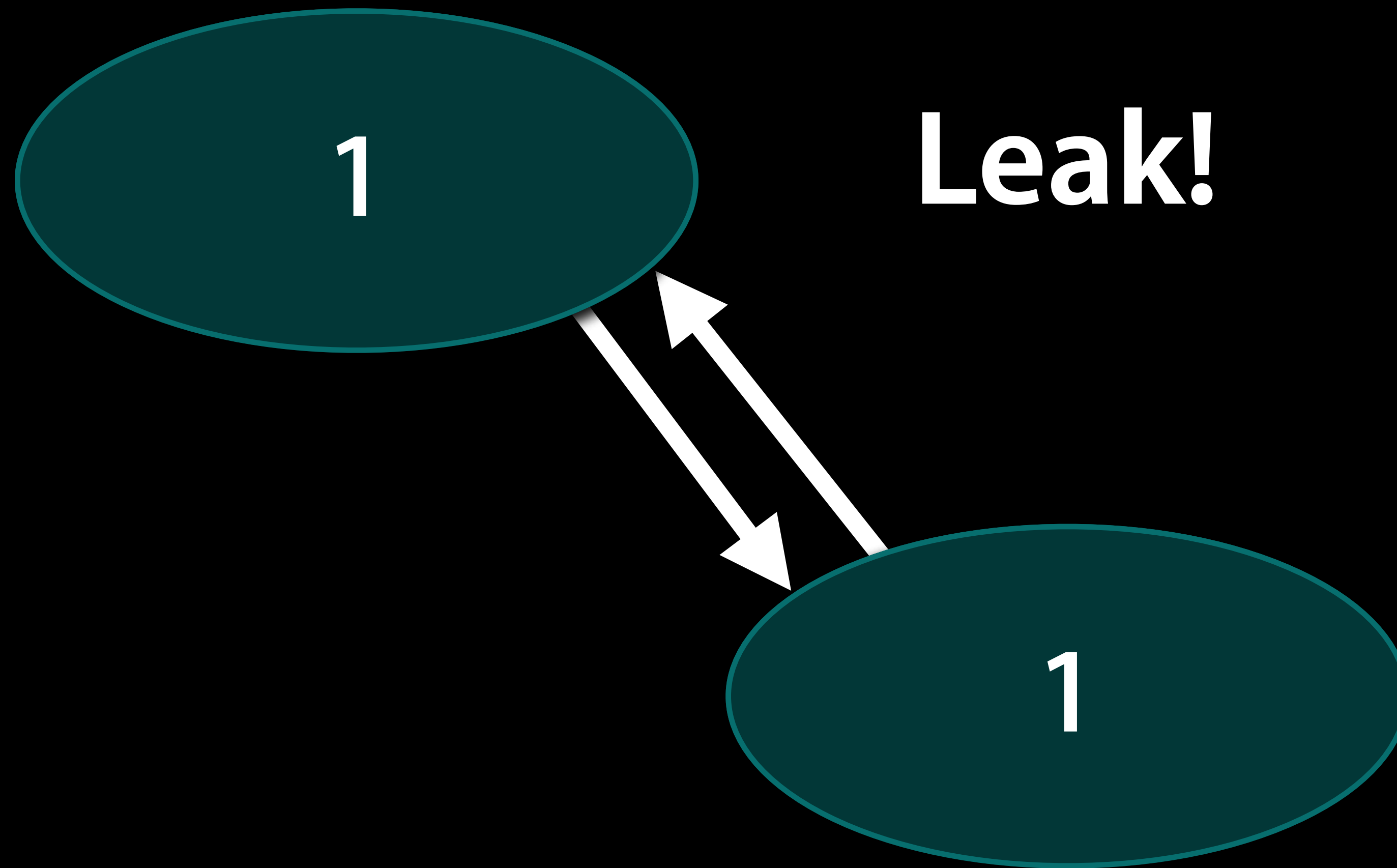


# Understanding Retain Cycles





# Understanding Retain Cycles



# Potential Retain Cycle Warning

```
- (void)example {  
    _ivar = ^{  
        [_ivar2 class];  
    };  
}
```

# Potential Retain Cycle Warning

```
- (void)example {  
    self->_ivar = ^{  
        [self->_ivar2 class];  
    };  
}
```

# Potential Retain Cycle Warning

```
- (void)example {  
    self->_ivar = ^{  
        [self->_ivar2 class];  
    };  
}
```

- The compiler implicitly references 'self'

# Potential Retain Cycle Warning

```
- (void)example {  
    _ivar = ^{  
        [_ivar2 class];  
    };  
}
```

# Potential Retain Cycle Warning

```
- (void)example {  
    _ivar = ^{  
        [_ivar2 class];  
    };  
}
```

**warning:** capturing 'self' strongly in this block is likely to lead to a retain cycle [-Warc-retain-cycles]

```
        [_ivar2 class];
```

~~~~~

note: block will be retained by an object strongly retained by the captured object

```
    _ivar = ^{
```

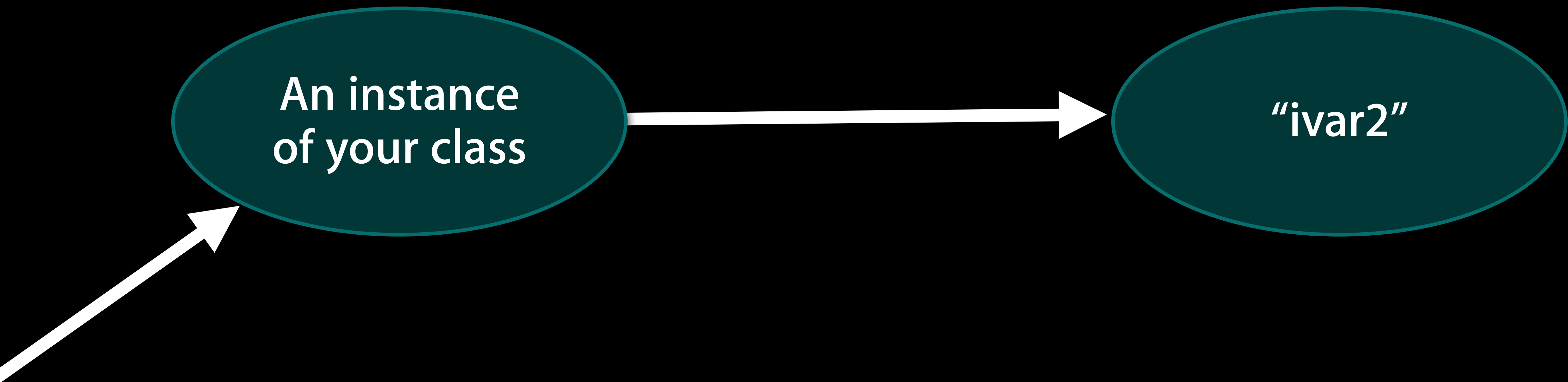
~~~~~

# Understanding Block Retain Cycles



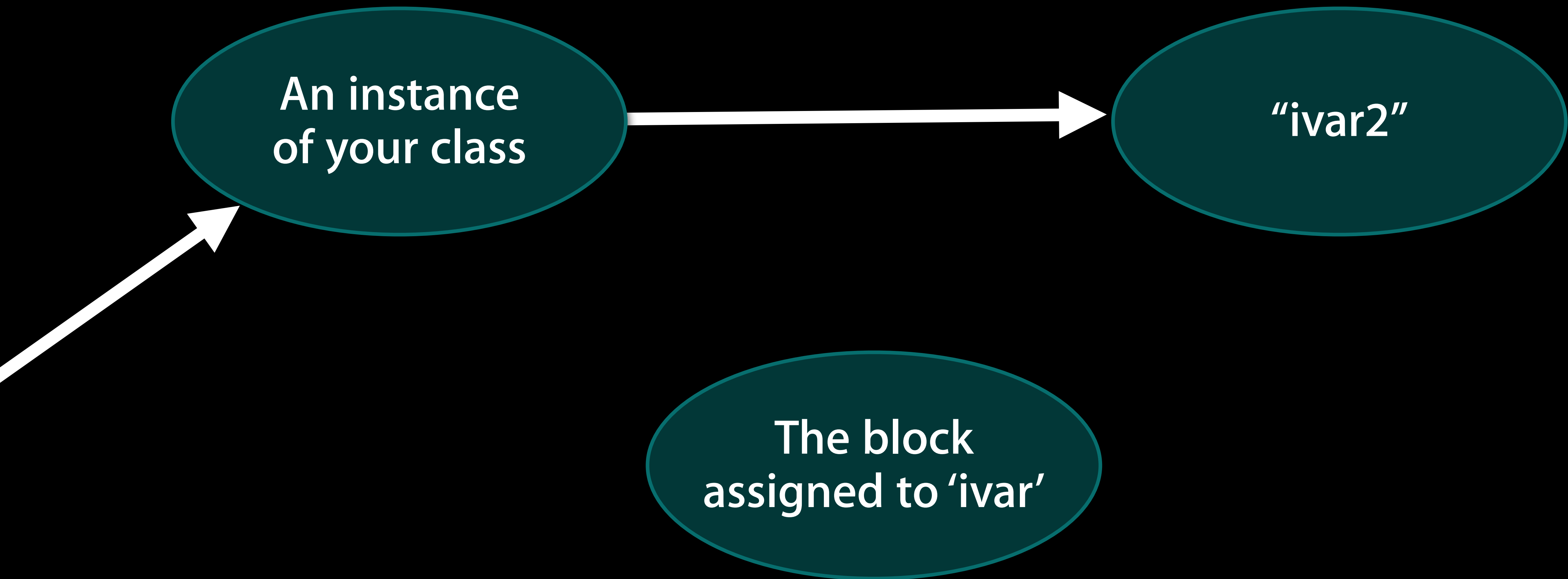
An instance  
of your class

# Understanding Block Retain Cycles

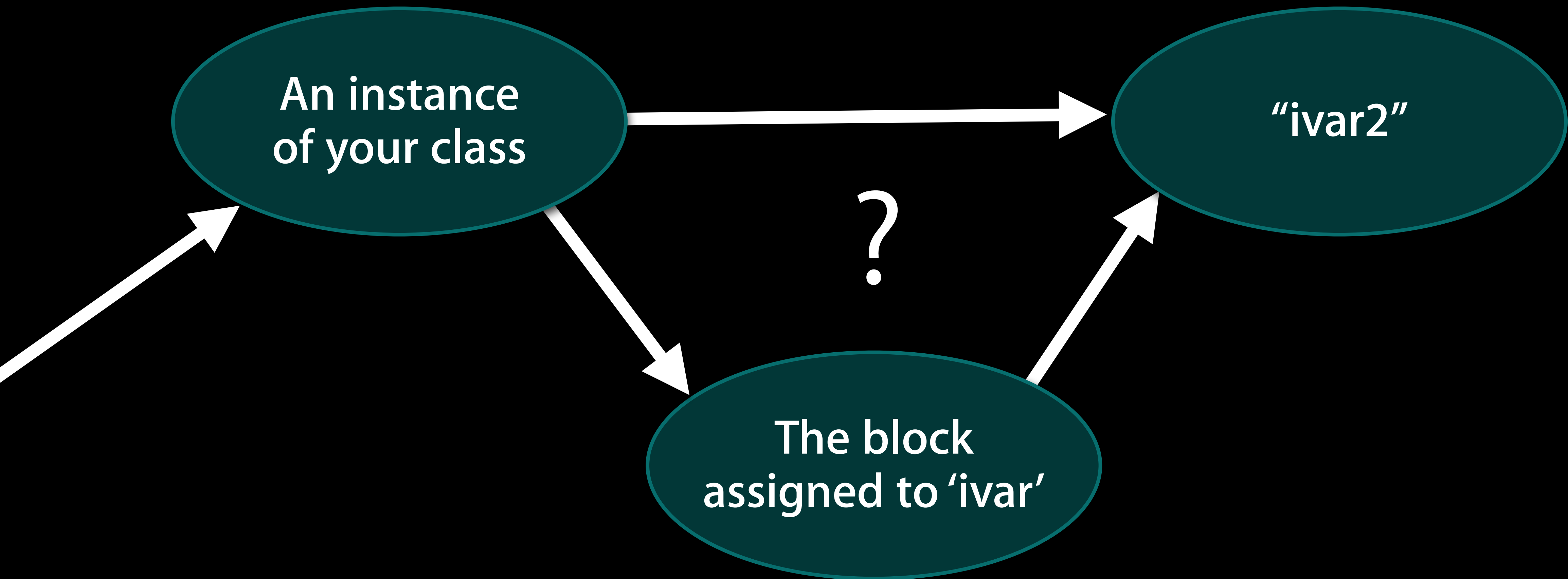




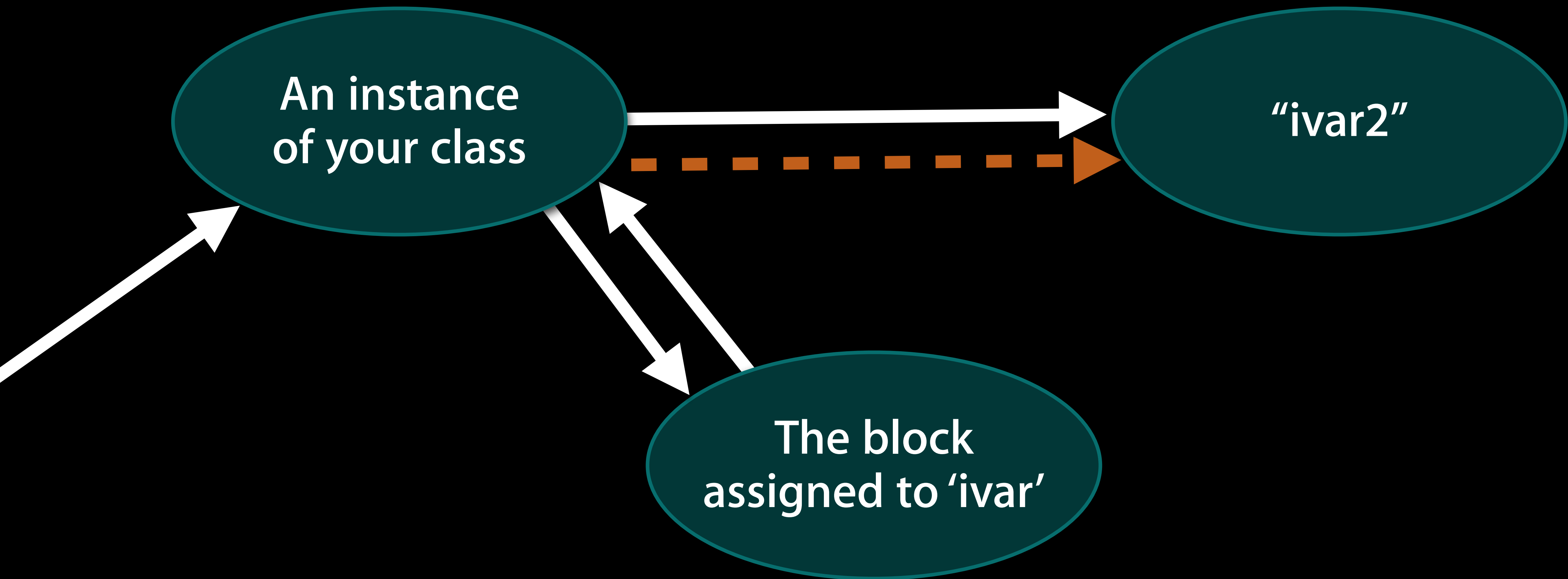
# Understanding Block Retain Cycles



# Understanding Block Retain Cycles



# Understanding Block Retain Cycles



# Fixing the Retain Cycle

```
- (void)example {  
    _ivar = ^{  
        [_ivar2 class];  
    };  
}
```

**warning:** capturing 'self' strongly in this block is likely to lead to a retain cycle [-Warc-retain-cycles]

```
    [_ivar2 class];
```

~~~~~

note: block will be retained by an object strongly retained by the captured object

```
    _ivar = ^{
```

~~~~~

# Fixing the Retain Cycle

```
- (void)example {  
    __weak MyClass *weak_self = self;  
    _ivar = ^{  
        [weak_self->_ivar2 class];  
    };  
}
```

**warning:** capturing 'self' strongly in this block is likely to lead to a retain cycle [-Warc-retain-cycles]

```
    [_ivar2 class];
```

~~~~~

note: block will be retained by an object strongly retained by the captured object

```
    _ivar = ^{
```

~~~~~

# Fixing the Retain Cycle

```
- (void)example {  
    __weak MyClass *weak_self = self;  
    _ivar = ^{  
        [weak_self->_ivar2 class];  
    };  
}
```

# Fixing the Retain Cycle

```
- (void)example {  
    __weak MyClass *weak_self = self;  
    _ivar = ^{  
        [weak_self->_ivar2 class];  
    };  
}
```

# Fixing the Retain Cycle

```
- (void)example {  
    __weak MyClass *weak_self = self;  
    _ivar = ^{  
        [weak_self->_ivar2 class];  
    };  
}
```

- Weak variables do not extend the lifetime of objects



# Fixing the Retain Cycle

```
- (void)example {  
    __weak MyClass *weak_self = self;  
    _ivar = ^{  
        [weak_self->_ivar2 class];  
    };  
}
```

- Weak variables do not extend the lifetime of objects
- Therefore they do not create retain cycles

# Fixing the Retain Cycle

```
- (void)example {  
    __weak MyClass *weak_self = self;  
    _ivar = ^{  
        [weak_self->_ivar2 class];  
    };  
}
```

- Weak variables do not extend the lifetime of objects
- Therefore they do not create retain cycles
- Weak variables safely become nil

# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

- Does this method get called?

# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

- Does this method get called?
- How do we reason about when 'weak\_ivar' is nil?

# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

**warning:** weak receiver may be unpredictably set to nil  
[-Wreceiver-is-weak]

# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

**warning:** weak receiver may be unpredictably set to nil  
[-Wreceiver-is-weak]

```
NSLog(@"%@", [_weak_ivar description]);
```



# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

**warning:** weak receiver may be unpredictably set to nil  
[-Wreceiver-is-weak]

```
    NSLog(@"%@", [_weak_ivar description]);
```



# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

**warning:** weak receiver may be unpredictably set to nil  
[-Wreceiver-is-weak]

```
    NSLog(@"%@", [_weak_ivar description]);  
                ^
```

**note:** assign the value to a strong variable to keep the object alive during use

# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

- Does this method get called zero, one, or two times?

# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

- Does this method get called zero, one, or two times?
- How do we reason about when 'weak\_ivar' is nil?

# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

**warning:** weak instance variable '\_weak\_ivar' is accessed multiple times in this method but may be unpredictably set to nil; assign to a strong variable to keep the object alive [-Warc-repeated-use-of-weak]

# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

**warning:** weak instance variable '\_weak\_ivar' is accessed multiple times in this method but may be unpredictably set to nil; assign to a strong variable to keep the object alive [-Warc-repeated-use-of-weak]

```
    NSLog(@"%@", [_weak_ivar description]);
```



# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

**warning:** weak instance variable '\_weak\_ivar' is accessed multiple times in this method but may be unpredictably set to nil; assign to a strong variable to keep the object alive [-Warc-repeated-use-of-weak]

```
    NSLog(@"%@", [_weak_ivar description]);
```

^~~~~~

# Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

**warning:** weak instance variable '\_weak\_ivar' is accessed multiple times in this method but may be unpredictably set to nil; assign to a strong variable to keep the object alive [-Warc-repeated-use-of-weak]

```
    NSLog(@"%@", [_weak_ivar description]);
```

~~~~~

Predictably Accessing Weak Variables

```
- (void)example {  
    NSLog(@"%@", [_weak_ivar description]);  
}
```

warning: weak receiver may be unpredictably set to nil
[-Wreceiver-is-weak]

```
    NSLog(@"%@", [_weak_ivar description]);  
                ^
```

note: assign the value to a strong variable to keep the object alive during use

Predictably Accessing Weak Variables

```
- (void)example {  
  
    NSLog(@"%@", [_weak_ivar description]);  
  
}
```

warning: weak receiver may be unpredictably set to nil
[-Wreceiver-is-weak]

```
    NSLog(@"%@", [_weak_ivar description]);
```



note: assign the value to a strong variable to keep the object alive during use

Predictably Accessing Weak Variables

```
- (void)example {  
    NSString *tmp = _weak_ivar;  
    if (tmp) {  
        NSLog(@"%@", [tmp description]);  
    }  
}
```

warning: weak receiver may be unpredictably set to nil
[-Wreceiver-is-weak]

```
    NSLog(@"%@", [_weak_ivar description]);
```

^

note: assign the value to a strong variable to keep the object alive during use

Predictably Accessing Weak Variables

```
- (void)example {  
    NSString *tmp = _weak_ivar;  
    if (tmp) {  
        NSLog(@"%@", [tmp description]);  
    }  
}
```

Predictably Accessing Weak Variables

```
- (void)example {  
    NSString *tmp = _weak_ivar;  
    if (tmp) {  
        NSLog(@"%@", [tmp description]);  
    }  
}
```

Predictably Accessing Weak Variables

```
- (void)example {  
    NSString *tmp = _weak_ivar;  
    if (tmp) {  
        NSLog(@"%@", [tmp description]);  
    }  
}
```

- “tmp” is valid for the scope of the ‘if’ block

Predictably Accessing Weak Variables

```
- (void)example {  
    NSString *tmp = _weak_ivar;  
    if (tmp) {  
        NSLog(@"%@", [tmp description]);  
    }  
}
```

- “tmp” is valid for the scope of the ‘if’ block
- Handling the “weak is nil” case is natural

Improving CoreFoundation and ARC

Improving CoreFoundation and ARC

```
NSString *string = (__bridge NSString *)CFDictionaryGetValue(_dict, @"key");
```

Improving CoreFoundation and ARC

```
NSString *string = (__bridge NSString *)CFDictionaryGetValue(_dict, @"key");
```

- The ARC compiler must reason about object lifetime

Improving CoreFoundation and ARC

```
NSString *string = (__bridge NSString *)CFDictionaryGetValue(_dict, @"key");
```

- The ARC compiler must reason about object lifetime
- Requires retain count “bridging” between in and out of ARC

Improving CoreFoundation and ARC

```
NSString *string = (__bridge NSString *)CFDictionaryGetValue(_dict, @"key");
```

- The ARC compiler must reason about object lifetime
- Requires retain count “bridging” between in and out of ARC
 - +1 via CFBridgingRetain()

Improving CoreFoundation and ARC

```
NSString *string = (__bridge NSString *)CFDictionaryGetValue(_dict, @"key");
```

- The ARC compiler must reason about object lifetime
- Requires retain count “bridging” between in and out of ARC
 - +1 via CFBridgingRetain()
 - -1 via CFBridgingRelease()

Improving CoreFoundation and ARC

```
NSString *string = (__bridge NSString *)CFDictionaryGetValue(_dict, @"key");
```

- The ARC compiler must reason about object lifetime
- Requires retain count “bridging” between in and out of ARC
 - +1 via CFBridgingRetain()
 - -1 via CFBridgingRelease()
 - +0 via “__bridge” casts to avoid mistakes

CoreFoundation Conventions

```
NSString *string = (__bridge NSString *)CFDictionaryGetValue(_dict, @"key");
```

CoreFoundation Conventions

```
NSString *string = (__bridge NSString *)CFDictionaryGetValue(_dict, @"key");
```

- Common CF functions have been audited
 - "...Create()" and "...Copy..." return +1
 - Everything else is +0

CoreFoundation Conventions

```
NSString *string = (__bridge NSString *)CFDictionaryGetValue(_dict, @"key");
```

- Common CF functions have been audited
 - "...Create()" and "...Copy..." return +1
 - Everything else is +0
- Compiler attributes for exceptions
 - CF_RETURNS_RETAINED and CF_RETURNS_NOT_RETAINED
 - CF_RELEASES_ARGUMENT

CoreFoundation Conventions

```
NSString *string = (__bridge NSString *)CFDictionaryGetValue(_dict, @"key");
```

- Common CF functions have been audited
 - "...Create()" and "...Copy..." return +1
 - Everything else is +0
- Compiler attributes for exceptions
 - CF_RETURNS_RETAINED and CF_RETURNS_NOT_RETAINED
 - CF_RELEASES_ARGUMENT
- These also help the static analyzer

Improving CoreFoundation and ARC

```
NSString *string = (__bridge NSString *)CFDictionaryGetValue(_dict, @"key");
```

Improving CoreFoundation and ARC

```
NSString *string = (__bridge NSString *)CFDictionaryGetValue(_dict, @"key");
```

- The “everything else” case is now formalized

Improving CoreFoundation and ARC

```
NSString *string = (__bridge NSString *)CFDictionaryGetValue(_dict, @"key");
```

- The “everything else” case is now formalized
- Common CF APIs allow implicit bridging

Improving CoreFoundation and ARC

```
NSString *string = CFDictionaryGetValue(_dict, @"key");
```

- The “everything else” case is now formalized
- Common CF APIs allow implicit bridging

Improving CoreFoundation and ARC

```
NSString *string = CFDictionaryGetValue(_dict, @"key");
```

- The “everything else” case is now formalized
- Common CF APIs allow implicit bridging
- New macros are available for your use too

Enabling Implicit Bridging

Enabling Implicit Bridging

```
#include <CoreFoundation/CoreFoundation.h>
```

```
EXArrayRef EXFooCreateCopy(...);  
const void *EXFooGetValueAtIndex(EXArrayRef theArray, CFIndex idx);  
const void *EXFooRandomPlusOne(EXArrayRef theArray);
```

Enabling Implicit Bridging

```
#include <CoreFoundation/CoreFoundation.h>
```

```
EXArrayRef EXFooCreateCopy(...); // GOOD: follows the naming convention  
const void *EXFooGetValueAtIndex(EXArrayRef theArray, CFIndex idx);  
const void *EXFooRandomPlusOne(EXArrayRef theArray);
```

Enabling Implicit Bridging

```
#include <CoreFoundation/CoreFoundation.h>
```

```
EXArrayRef EXFooCreateCopy(...);  
const void *EXFooGetValueAtIndex(EXArrayRef theArray, CFIndex idx);  
const void *EXFooRandomPlusOne(EXArrayRef theArray);
```

Enabling Implicit Bridging

```
#include <CoreFoundation/CoreFoundation.h>
```

```
EXArrayRef EXFooCreateCopy(...);  
const void *EXFooGetValueAtIndex(EXArrayRef theArray, CFIndex idx);  
const void *EXFooRandomPlusOne(EXArrayRef theArray) CF_RETURNS_RETAINED;
```

Enabling Implicit Bridging

```
#include <CoreFoundation/CoreFoundation.h>
```

```
CF_IMPLICIT_BRIDGING_ENABLED
```

```
EXArrayRef EXFooCreateCopy(...);  
const void *EXFooGetValueAtIndex(EXArrayRef theArray, CFIndex idx);  
const void *EXFooRandomPlusOne(EXArrayRef theArray) CF_RETURNS_RETAINED;
```

```
CF_IMPLICIT_BRIDGING_DISABLED
```

Enabling Implicit Bridging

```
#include <CoreFoundation/CoreFoundation.h>
```

```
// must be after all #includes / #imports
```

```
CF_IMPLICIT_BRIDGING_ENABLED
```

```
EXArrayRef EXFooCreateCopy(...);
```

```
const void *EXFooGetValueAtIndex(EXArrayRef theArray, CFIndex idx);
```

```
const void *EXFooRandomPlusOne(EXArrayRef theArray) CF_RETURNS_RETAINED;
```

```
CF_IMPLICIT_BRIDGING_DISABLED
```


Enabling Implicit Bridging

```
#include <CoreFoundation/CoreFoundation.h>
// explicitly bridged code

// must be after all #includes / #imports
CF_IMPLICIT_BRIDGING_ENABLED

EXArrayRef EXFooCreateCopy(...);
const void *EXFooGetValueAtIndex(EXArrayRef theArray, CFIndex idx);
const void *EXFooRandomPlusOne(EXArrayRef theArray) CF_RETURNS_RETAINED;

CF_IMPLICIT_BRIDGING_DISABLED

// explicitly bridged code
```

Wrap Up

Summary

- Modules
- Improved productivity
 - Better compiler warnings
- ARC
 - Faster, easier, safer

More Information

Dave DeLong

Developer Tools Evangelist
delong@apple.com

Documentation

Developer Tools Portal
<http://developer.apple.com/xcode>

Apple Developer Forums

<http://devforums.apple.com>

Related Sessions

What's New in the LLVM Compiler	Pacific Heights Tuesday 2:00PM	
Optimize Your Code Using LLVM	Nob Hill Wednesday 3:15PM	

Labs

Objective-C and LLVM

Tools Lab B
Wednesday 9AM

Objective-C and LLVM

Tools Lab C
Thursday 2PM

 WWDC2013