What's New in the LLVM Compiler

Session 402

Evan Cheng
Sr. Manager, Compilation Technologies

Focused on Providing Best-in-Class Tools

Focused on Providing Best-in-Class Tools

- Support for latest hardware

Focused on Providing Best-in-Class Tools

- Support for latest hardware
- Improving performance

Focused on Providing Best-in-Class Tools

- Support for latest hardware
- Improving performance
- Improving developer productivity

Support for Latest Hardware

armv7s Architecture

- Architecture for Apple A6 processor
  - iPhone 5 and new iPads
- Extensive tuning and optimization in the compiler
  - Uses instructions only available in armv7s

armv7s Architecture

- Architecture for Apple A6 processor
  - iPhone 5 and new iPads
- Extensive tuning and optimization in the compiler
  - Uses instructions only available in armv7s

Important for achieving max performance!

armv7s Architecture

- Already part of the standard architectures for iOS apps

Intel AVX

- 256-bit floating-point vector computation
  - Twice as wide as SSE vectors
  - Supported in Sandy Bridge and Ivy Bridge processors
- Good for loops with operations that can be performed in parallel
  - Floating-point intensive
  - High ratio of computation to memory bandwidth

Intel AVX2

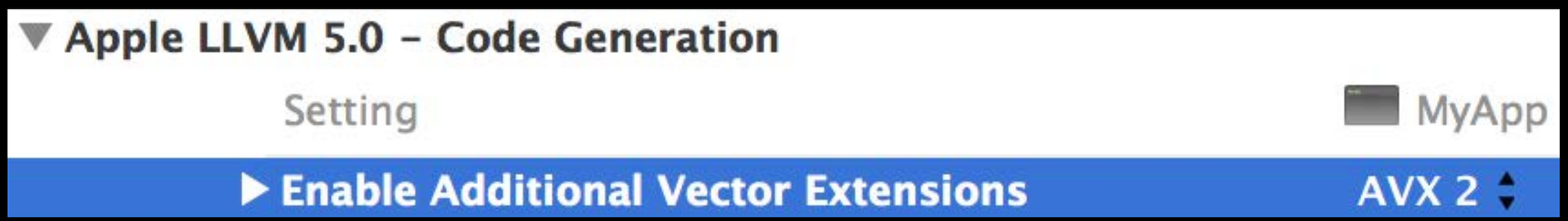
- Supported in "Haswell" processors
  - Extend the AVX instruction set to integers
  - Adds fused multiply-accumulate for increased floating point throughput
  - More extensive set of vector shuffle instructions

Using AVX2 with Fallback to AVX / SSE

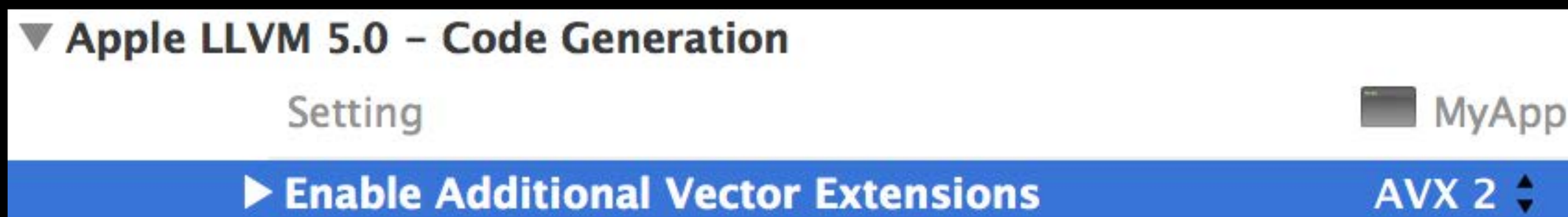
- Check at runtime if AVX2 is supported
- Put AVX2 code in separate files to be compiled with `-mavx2` option
- Provide an alternate version using AVX or SSE

```
#include <sys/sysctl.h>
void add(int size, int *in1, int *in2, int *out) {
  int answer = 0;
  size_t length = sizeof(answer);
  if (!sysctlbyname("hw.optional.avx2_0", &answer, &length, NULL, 0) && answer != 0)
    addAVX2(size, in1, in2, out);
  else if (!sysctlbyname("hw.optional.avx1_0", &answer, &length, NULL, 0) &&
           answer != 0)
    addAVX(size, in1, in2, out);
  else
    addSSE(size, in1, in2, out);
}
```

Intel AVX2 in Xcode 5



Intel AVX2 in Xcode 5



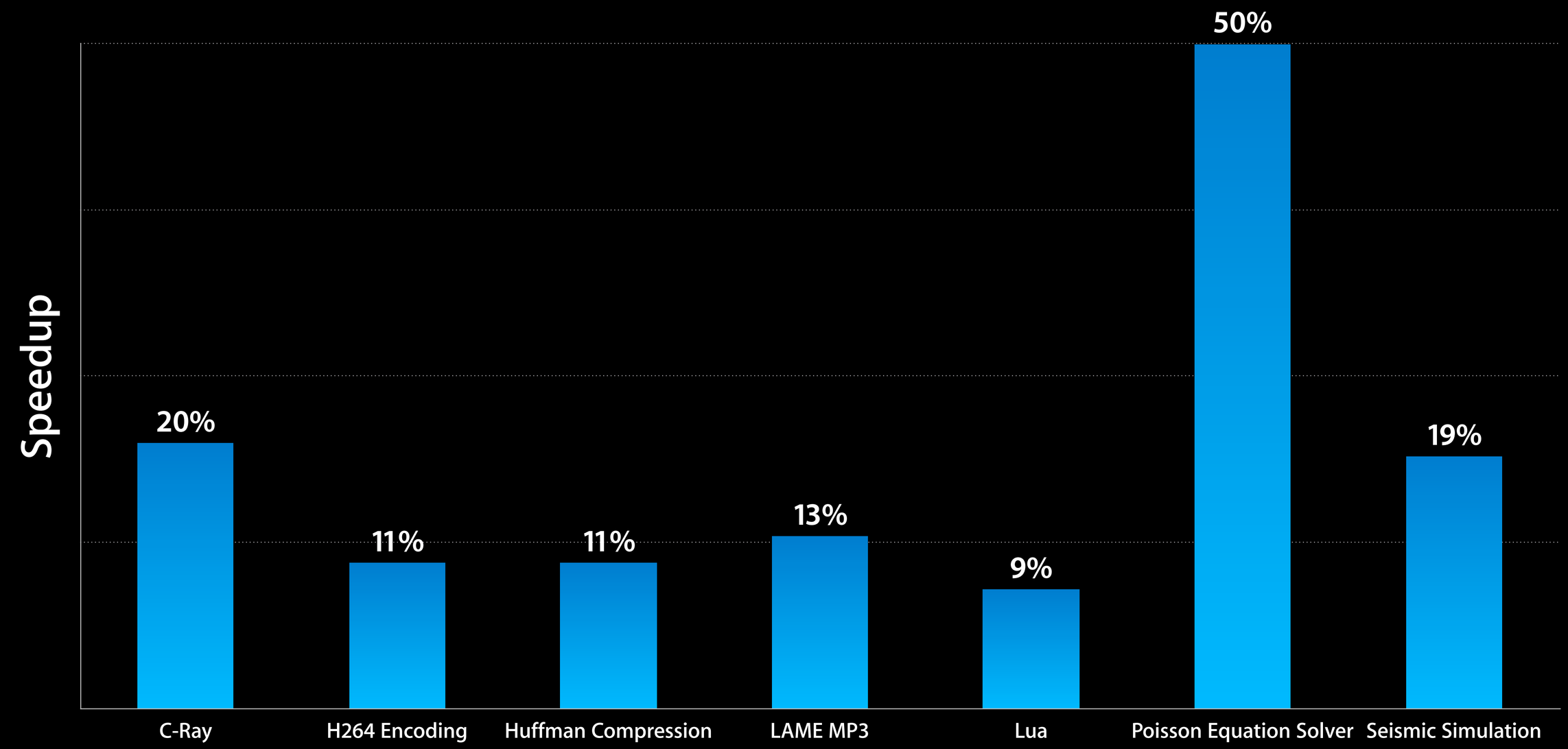
Performance

Performance Numbers

OS X

Performance Numbers

OS X

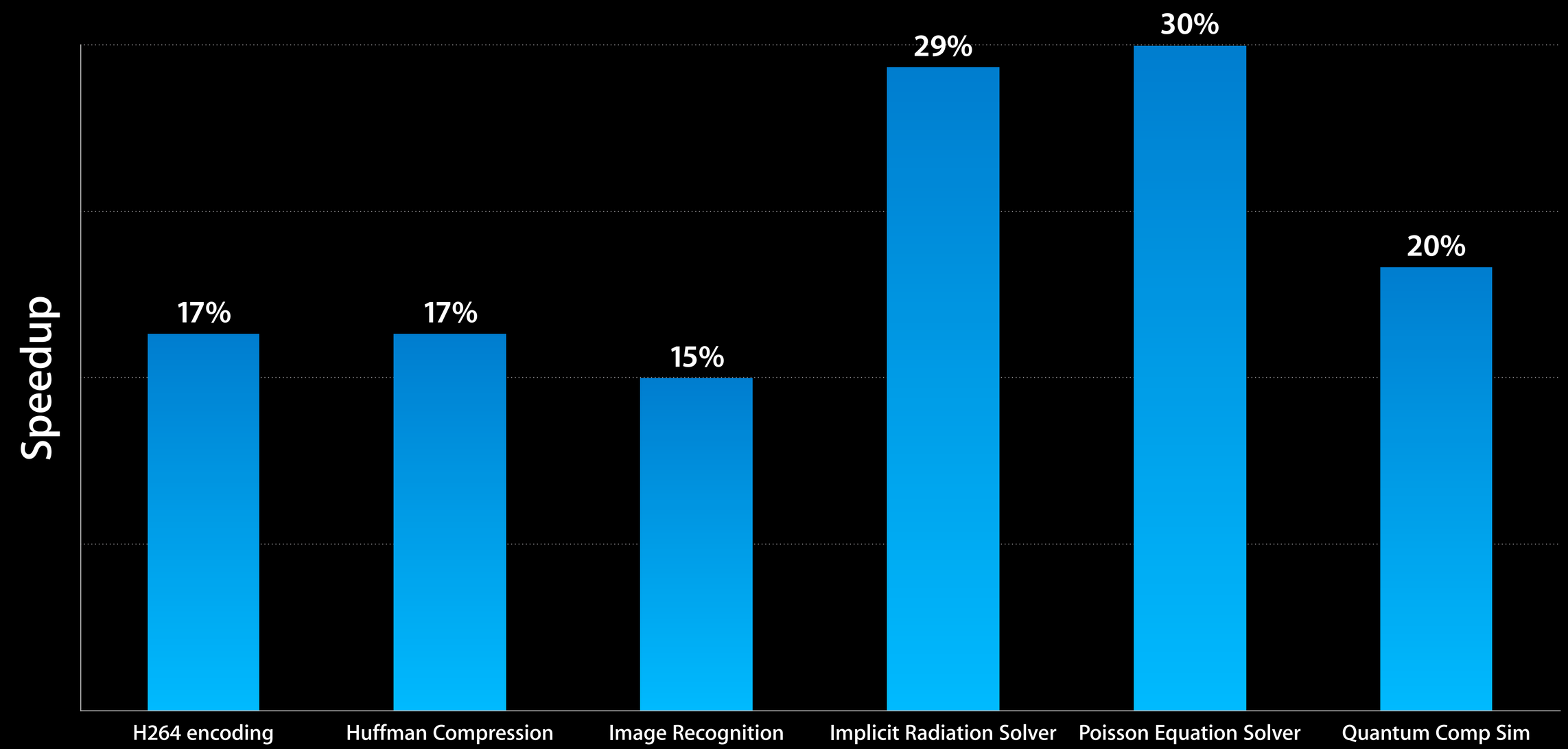


Performance Numbers

iOS

Performance Numbers

iOS



Strict Aliasing Enabled by Default

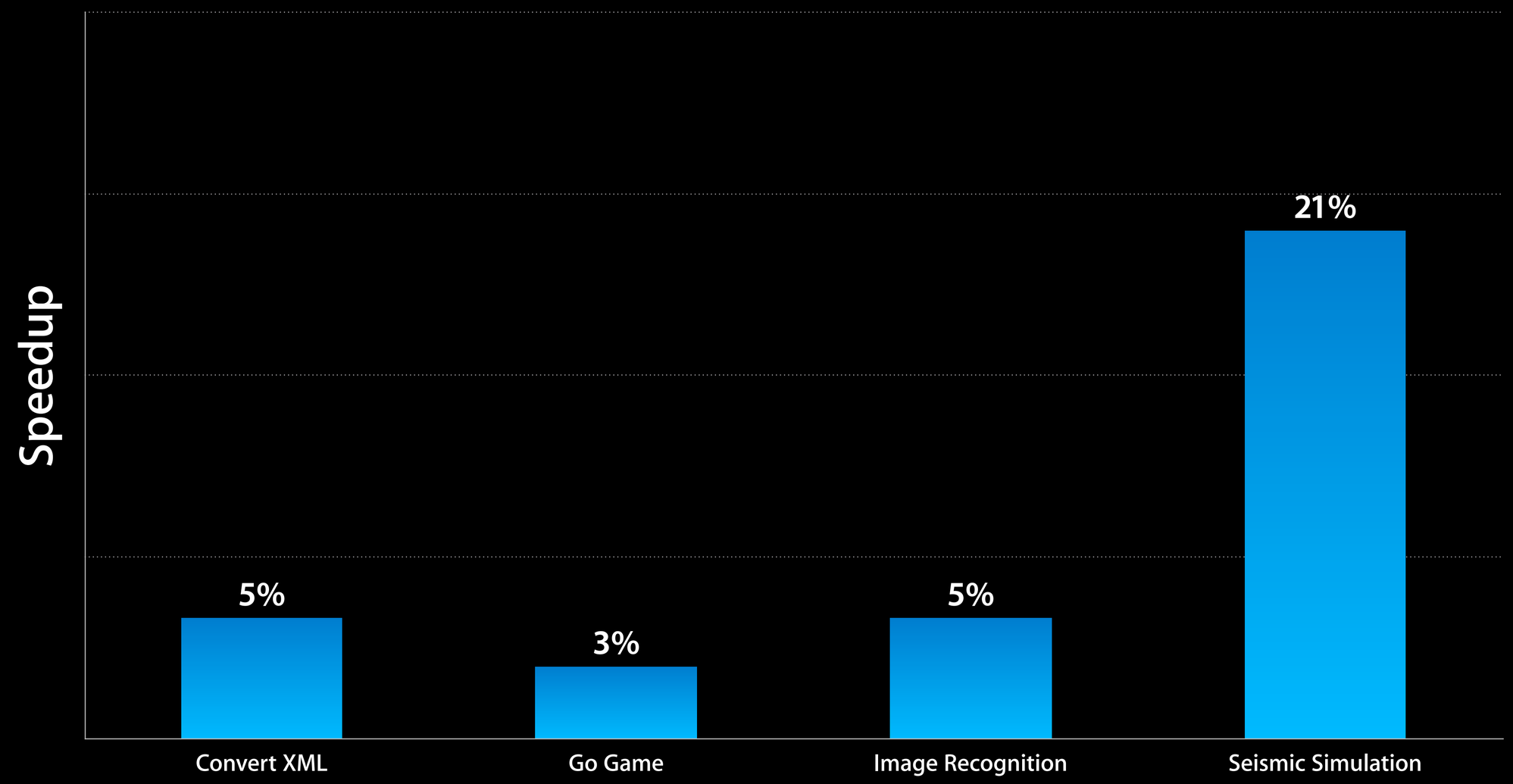
- Stronger alias analysis to enable more aggressive optimizations
- Enabled by default in Xcode 4.6

Strict Aliasing

Performance

Strict Aliasing

Performance



Strict Aliasing

Safety

- Do not use invalid pointer casts

```
// Little-endian layout.
struct Components {
 uint16_t red;
 uint16_t green;
 uint16_t blue;
 uint16_t alpha;
};

uint64_t color = UINT64_C(0xffff820005000500);
struct Components *components = &color;
…
// e.g., zero out the green component.
components->green = 0;
```

Strict Aliasing

Safety

- Use union and do not use pointers

```
union ColorComponents {
 uint64_t color;

 // Little-endian layout.
 struct {
   uint16_t red;
   uint16_t green;
   uint16_t blue;
   uint16_t alpha;
 } components;
};

union ColorComponents c = UINT64_C(0xffff820005000500);
…
// e.g., zero out the green component.
c.components.green = 0;
```

Link-time Optimization (LTO)

- Whole application optimization performed at link time
- May significantly improve the performance of your code
- Widely deployed at Apple

Link-time Optimization (LTO)

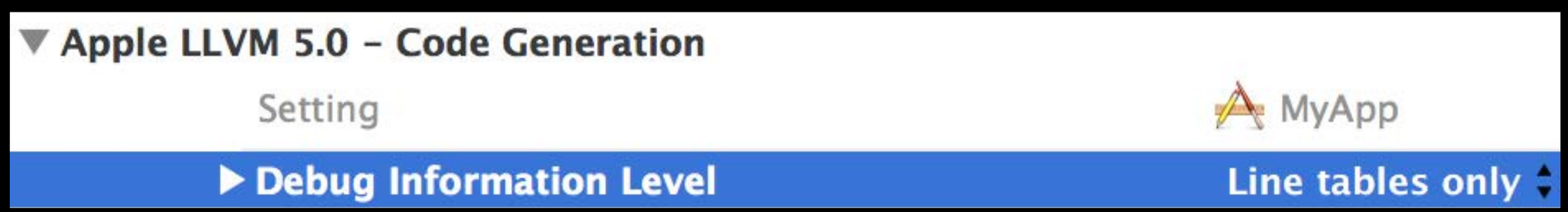
Performance wins

- Apple is now utilizing this technology to build our own products
  - Apple LLVM Compiler
    - Up to 6% faster
  - iOS Kernel
    - Up to 20% faster on certain file system operations
  - iOS iMovie app
    - Reduced binary size by 25%

Link-time Optimization (LTO)

Disclaimer

- May require too much memory for large C++ projects
- Try -gline-tables-only



Auto Vectorizer

- New in Xcode 5
- Accelerate some computation intensive loop automatically
- For both OS X and iOS applications

Auto Vectorizer

- New in Xcode 5
- Accelerate some computation intensive loop automatically
- For both OS X and iOS applications

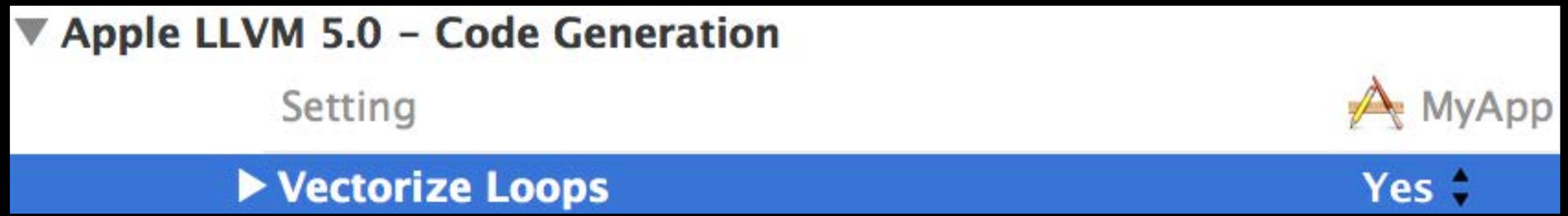
```
#include <arm_neon.h>
void increment_16_uint32(uint32x4_t *A) {
  uint32x4_t vec1 = vmovq_n_u32(1);
  for (int i = 0; i < 4; ++i) {
    *A = vaddq_u32(*A, vec1);
    ++A;
  }
}
```

Auto Vectorizer

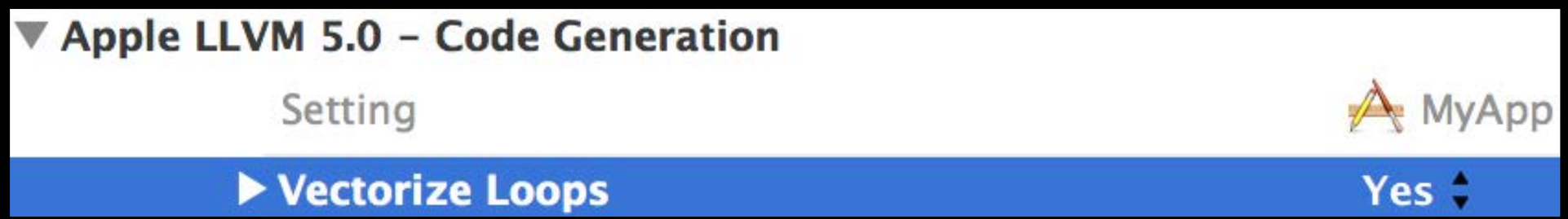
- New in Xcode 5
- Accelerate some computation intensive loop automatically
- For both OS X and iOS applications

```
void increment_16_uint32(unsigned *A) {
  for (int i = 0; i < 16; ++i)
    A[i] += 1;
}
```

Auto Vectorizer in Xcode 5



Auto Vectorizer in Xcode 5



Optimize Your Code Using LLVM

Nob Hill
Wednesday 3:15PM

New -Ofast Optimization Level

What is it?

New -Ofast Optimization Level

What is it?

New -Ofast Optimization Level

What is it?

- All -O3 optimizations
- Enable much improved -ffast-math
- Enables the vectorizer
- Makes it easy to get maximum optimization

New -Ofast Optimization Level

What is it?

- All -O3 optimizations
- Enable much improved -ffast-math
- Enables the vectorizer
- Makes it easy to get maximum optimization

Apple LLVM 5.0 – Code Generation	
Setting	MyApp
Optimization Level	<Multiple values>
Debug	None [-O0]
Release	Fastest, Aggressive Optimizations [-Ofast]

New -Ofast Optimization Level

Disclaimer

- Do not use if your application has high level floating point precision requirement
- Test carefully
- Does not enable LTO

New -Ofast Optimization Level

Disclaimer

- Do not use if your application has high level floating point precision requirement
- Test carefully
- Does not enable LTO

Improving Developer Productivity

Bob Wilson
Manager, LLVM Core Team

Productivity Enhancements

- Compiler and tool updates
- C++ updates
- Compiler warnings
- Static analyzer
- Getting more out of your comments

Compiler and Tool Updates

Compiler Transition Completed

Compiler Transition Completed

Compiler Transition Completed

- Xcode 5 no longer includes gcc or llvm-gcc

Compiler Transition Completed

- Xcode 5 no longer includes gcc or llvm-gcc
- Apple now has only one compiler
  - Consistent with Xcode

Compiler Transition Completed

- Xcode 5 no longer includes gcc or llvm-gcc
- Apple now has only one compiler
  - Consistent with Xcode
- Focus on further advances in LLVM…

Command Line Tools

Command Line Tools

- Used to build common Unix software

Command Line Tools

- Used to build common Unix software
- Two components:

Tools

/usr/bin

OS X SDK

/usr/include
/usr/lib
/System/Library/Frameworks

OS X 10.9: Can Use Xcode

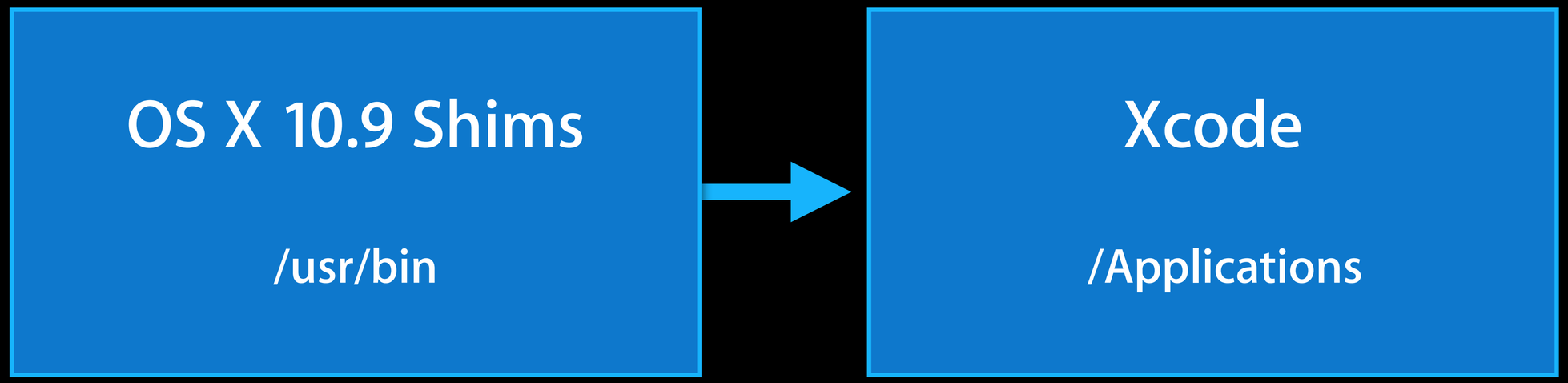
NEW

OS X 10.9: Can Use Xcode

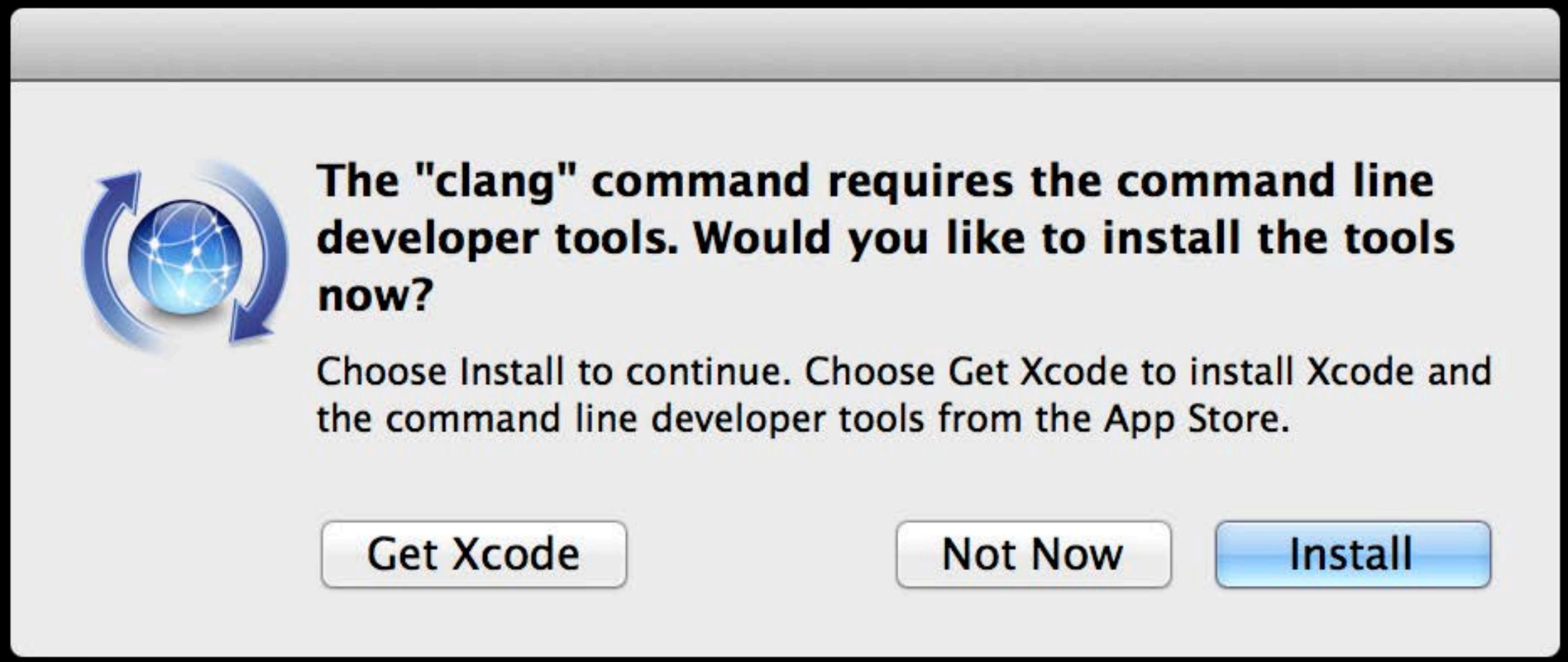
- Xcode already has everything you need

OS X 10.9: Can Use Xcode

- Xcode already has everything you need
- OS X 10.9 has shims for tools in `/usr/bin`

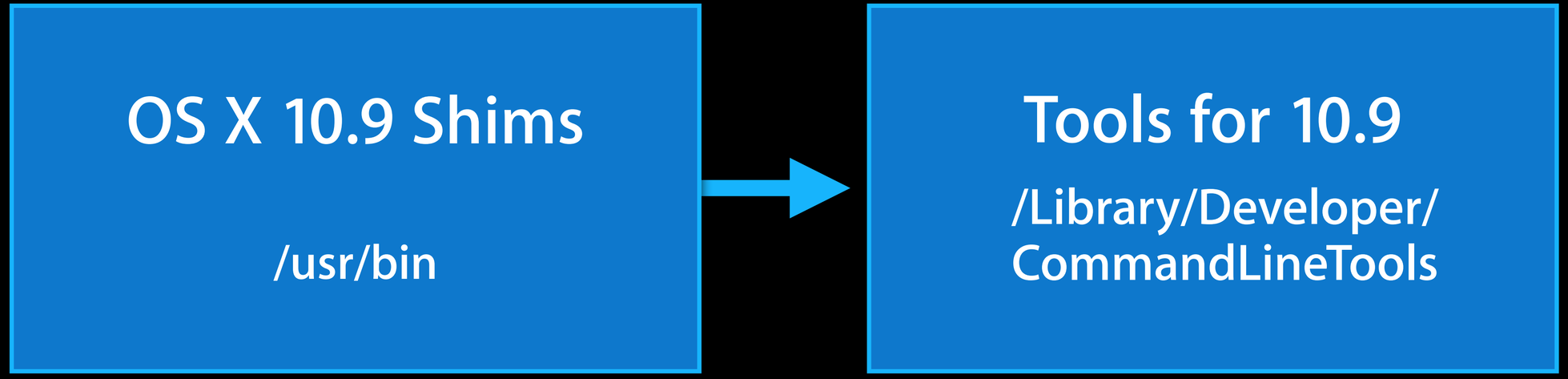


What if You Don’t Have Xcode?



New Command Line Tools for OS X 10.9

- Shims can forward to standalone tools instead of Xcode
- Software Update notifies you when new versions are available
- Easily removed: all files in one place



SDK Files Moved

SDK Files Moved

- SDK from Command Line Tools not in /

SDK Files Moved

- SDK from Command Line Tools not in /
- Compiler knows where to find default OS X SDK

SDK Files Moved

- SDK from Command Line Tools not in `/`
- Compiler knows where to find default OS X SDK
- Avoid hardcoded references to SDK files
  - `/usr/include`
  - `/usr/lib`
  - `/System/Library/Frameworks`

Command Line Tips

Command Line Tips

- xcrun now included with Command Line Tools

Command Line Tips

- xcrun now included with Command Line Tools
- To find the SDK:

```
xcrun --show-sdk-path --sdk macosx
```

Command Line Tips

- xcrun now included with Command Line Tools
- To find the SDK:

```
xcrun --show-sdk-path --sdk macosx
```

- No need for `-isysroot` compiler option with xcrun

```
xcrun --sdk iphoneos clang -c MyApp.c
```

Command Line Tips

- xcrun now included with Command Line Tools
- To find the SDK:

```
xcrun --show-sdk-path --sdk macosx
```

- No need for `-isysroot` compiler option with xcrun

```
xcrun --sdk iphoneos clang -c MyApp.c
```

- xcrun also pays attention to `SDKROOT` environment variable

C++ Updates

C++11 Update

C++11 Update

- Default for C++ in new projects since Xcode 4.4
- Almost complete support in Apple LLVM 5.0

C++11 Update

- Default for C++ in new projects since Xcode 4.4
- Almost complete support in Apple LLVM 5.0
- New features added this year:

Atomics and Memory Model	**Alignment Support**	**Inheriting Constructors**	**Generalized Attributes**	**Sequence Points**

C++11 Update

- Default for C++ in new projects since Xcode 4.4
- Almost complete support in Apple LLVM 5.0
- New features added this year:

Atomics and Memory Model	Alignment Support	Inheriting Constructors	Generalized Attributes	Sequence Points

Inheriting Constructors


```
class X {
public:
  X() : a(1), b(2.0) { }
  X(int a) : a(a), b(2.0) { }
  X(float b) : a(1), b(b) { }
  X(int a, float b) : a(a), b(b) { }
private:
  int a;
  float b;
};
```

Inheriting Constructors


```
class X {
public:
  X() : a(1), b(2.0) { }
  X(int a) : a(a), b(2.0) { }
  X(float b) : a(1), b(b) { }
  X(int a, float b) : a(a), b(b) { }
private:
  int a;
  float b;
};
```

```
class Y : X {
public:
  Y() : X() { }
  Y(int a) : X(a) { }
  Y(float b) : X(b) { }
  Y(int a, float b) : X(a, b) { }
private:
  int c;
};
```

- Lots of boilerplate to delegate to the base class constructors
- Changing the base class is error prone
- Inheriting constructors: implicitly declare forwarding constructors

Inheriting Constructors


```
class X {
public:
  X() : a(1), b(2.0) { }
  X(int a) : a(a), b(2.0) { }
  X(float b) : a(1), b(b) { }
  X(int a, float b) : a(a), b(b) { }
private:
  int a;
  float b;
};
```

```
class Y : X {
public:
  using X::X;



private:
  int c;
};
```

- Lots of boilerplate to delegate to the base class constructors
- Changing the base class is error prone
- Inheriting constructors: implicitly declare forwarding constructors

Non-Static Data Member Initializers

```
class X {
public:
  X() : a(1), b(2.0) { }
  X(int a) : a(a), b(2.0) { }
  X(float b) : a(1), b(b) { }
  X(int a, float b) : a(a), b(b) { }
private:
  int a;
  float b;
};
```

```
class Y : X {
public:
  using X::X;



private:
  int c = 3;
};
```

- Data member initializers used *unless* constructor provides an initializer

Non-Static Data Member Initializers

```
class X {
public:
  X() : a(1), b(2.0) { }
  X(int a) : a(a), b(2.0) { }
  X(float b) : a(1), b(b) { }
  X(int a, float b) : a(a), b(b) { }
private:
  int a = 1;
  float b = 2.0;
};
```

```
class Y : X {
public:
  using X::X;



private:
  int c = 3;
};
```

- Data member initializers used *unless* constructor provides an initializer

Non-Static Data Member Initializers

```
class X {
public:
  X() { }
  X(int a) : a(a) { }
  X(float b) : b(b) { }
  X(int a, float b) : a(a), b(b) { }
private:
  int a = 1;
  float b = 2.0;
};
```

```
class Y : X {
public:
  using X::X;



private:
  int c = 3;
};
```

- Data member initializers used *unless* constructor provides an initializer

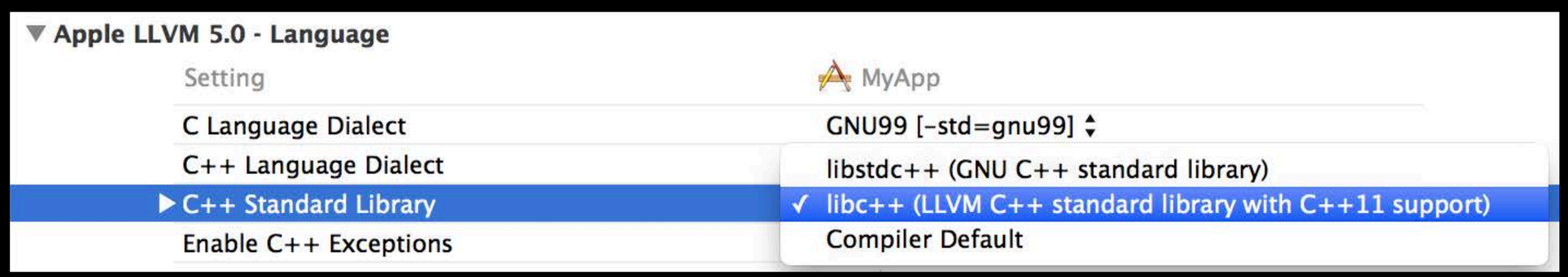
libc++ Transition

libc++ Transition

- LLVM C++ Standard Library
  - Required for many C++11 language features
  - Provides new C++11 library components

libc++ Transition

- LLVM C++ Standard Library
  - Required for many C++11 language features
  - Provides new C++11 library components
- Now the default for iOS 7 and OS X 10.9
  - Already the default for new projects
  - Can deploy back to iOS 5 or OS X 10.7



Compiler Warnings

Better Warnings → Fewer Bugs

- Apple LLVM compiler helps catch bugs before they bite
- Recent improvements:
  - New compiler warnings
  - More warnings enabled by default
  - Serious problems treated as errors

Unsequenced Modifications

- Warns about non-portable code
  - Order of some operations is not specified
  - LLVM may behave differently than other compilers
- New warning is enabled by default

Unsequenced Modifications

- Warns about non-portable code
  - Order of some operations is not specified
  - LLVM may behave differently than other compilers
- New warning is enabled by default

```
int Increment(int x) {
  x = x++;
  return x;
}
```

Unsequenced Modifications

- Warns about non-portable code
  - Order of some operations is not specified
  - LLVM may behave differently than other compilers
- New warning is enabled by default

```
int Increment(int x) {
  x = x++;
  return x;
}
```

```
warning: multiple unsequenced modifications to 'x' [-Wunsequenced]
  x = x++;
    ~  ^
```

Integer Overflow

- Warns about overflow in integer calculations
- New warning is enabled by default

Integer Overflow

- Warns about overflow in integer calculations
- New warning is enabled by default

```
int MultiplyConstants(void) {
  return 123456 * 789012;
}
```

Integer Overflow

- Warns about overflow in integer calculations
- New warning is enabled by default

```
int MultiplyConstants(void) {
  return 123456 * 789012;
}
```

```
warning: overflow in expression;
         result is -1375982336 with type 'int' [-Winteger-overflow]
  return 123456 * 789012;
                ^
```

Integer Overflow

- Warns about overflow in integer calculations
- New warning is enabled by default

```
long long MultiplyConstants(void) {
  return 123456LL * 789012LL;
}
```

Unused Functions

- Warns about dead code that can be removed
- Now enabled by default in new projects

Unused Functions

- Warns about dead code that can be removed
- Now enabled by default in new projects

```
static int Leftover(void);
static int PlusOne(int x) { return x+1; }
```

Unused Functions

- Warns about dead code that can be removed
- Now enabled by default in new projects

```
static int Leftover(void);
static int PlusOne(int x) { return x+1; }
```

```
warning: unused function 'Leftover' [-Wunused-function]
static int Leftover(void);
           ^
warning: unused function 'PlusOne' [-Wunused-function]
static int PlusOne(int x) { return x+1; }
           ^
```

Implicit Boolean Conversions

- Warns about implicit conversions to boolean values in C++
- Enabled by default in new projects

Implicit Boolean Conversions

- Warns about implicit conversions to boolean values in C++
- Enabled by default in new projects

```
extern int helper();
int BadConversion() {
  if (helper) return helper();
  return 0;
}
```

Implicit Boolean Conversions

- Warns about implicit conversions to boolean values in C++
- Enabled by default in new projects

```
extern int helper();
int BadConversion() {
  if (helper) return helper();
  return 0;
}
```

```
warning: address of function 'helper' will always evaluate to 'true'
         [-Wbool-conversion]
note: prefix with the address-of operator to silence this warning
  if (helper) return helper();
      ^
      &
```

Implicit Boolean Conversions

- Warns about implicit conversions to boolean values in C++
- Enabled by default in new projects

```
extern __attribute__((weak)) int helper();
int BadConversion() {
  if (helper) return helper();
  return 0;
}
```

Implicit Enum Conversions

- Warns about implicit conversions to enum types
- Enabled by default in new projects

Implicit Enum Conversions

- Warns about implicit conversions to enum types
- Enabled by default in new projects

```
typedef NS_ENUM(NSInteger, Shapes) { Circle, Square, Triangle };
typedef NS_ENUM(NSInteger, Colors) { Red, Yellow, Blue };
void Draw(Shapes S, Colors C);
void BadConversion() {
  Draw(Blue, Circle);
}
```

Implicit Enum Conversions

- Warns about implicit conversions to enum types
- Enabled by default in new projects

```
typedef NS_ENUM(NSInteger, Shapes) { Circle, Square, Triangle };
typedef NS_ENUM(NSInteger, Colors) { Red, Yellow, Blue };
void Draw(Shapes S, Colors C);
void BadConversion() {
  Draw(Blue, Circle);
}
```

```
warning: implicit conversion from enumeration type 'enum Colors'
         to different enumeration type 'Shapes' (aka 'enum Shapes')
         [-Wenum-conversion]
  Draw(Blue, Circle);
  ~~~~ ^~~~
```

Undeclared Selectors

- Warns about "@selector(…)" expression with undeclared selector
- Enabled by default in new projects

Undeclared Selectors

- Warns about "@selector(…)" expression with undeclared selector
- Enabled by default in new projects

```
[NSTimer scheduledTimerWithTimeInterval:60.0 target:self
 selector:@selector(cloze)
 userInfo:nil repeats:NO];
```

Undeclared Selectors

- Warns about "@selector(…)" expression with undeclared selector
- Enabled by default in new projects

```
[NSTimer scheduledTimerWithTimeInterval:60.0 target:self
 selector:@selector(cloze)
 userInfo:nil repeats:NO];
```

```
warning: undeclared selector 'cloze' [-Wundeclared-selector]
  selector:@selector(cloze)
           ^
```

Mismatched Return Types

- Compiler detects missing return values
- Now treated as an error by default

Mismatched Return Types

- Compiler detects missing return values
- Now treated as an error by default

```
float SafeSqrt(float f) {
  if (f < 0) return;
  return sqrtf(f);
}
```

Mismatched Return Types

- Compiler detects missing return values
- Now treated as an error by default

```
float SafeSqrt(float f) {
  if (f < 0) return;
  return sqrtf(f);
}
```

```
error: non-void function 'SafeSqrt' should return a value [-Wreturn-type]
  if (f < 0) return;
             ^
```

Unintentional Root Classes

```
@interface MyClass
@end
```

Unintentional Root Classes

```
@interface MyClass
@end
```

```
error: class 'MyClass' defined without specifying a base class
       [-Werror,-Wobjc-root-class]
note: add a super class to fix this problem
@interface MyClass
                  ^
                  : NSObject
```

Unintentional Root Classes

```
@interface MyClass
@end
```

```
error: class 'MyClass' defined without specifying a base class
       [-Werror,-Wobjc-root-class]
note: add a super class to fix this problem
@interface MyClass
                  ^
                  : NSObject
```

- Now defaults to an error for new projects
- Can add `NS_ROOT_CLASS` before `@interface`

Warning Summary

	New	Enabled by Default	Enabled in New Projects	Errors in New Projects
Unsequenced modifications	Yes	Yes	Yes	
Integer overflow	Yes	Yes	Yes	
Unused functions			Yes	
Implicit boolean conversions			Yes	
Implicit enum conversions			Yes	
Undeclared selectors			Yes	
Mismatched return types			Yes	Yes
Unintentional root classes		Yes	Yes	Yes

Productivity Enhancements

- Compiler and tool updates
- C++ updates
- Compiler warnings
- Static analyzer
- Getting more out of your comments

Static Analyzer

Anna Zaks
Engineer, Compiler Frontend Team

Bugs are Bad!

Why Use the Static Analyzer?

- Performs deeper code analysis than a compiler
- Systematically explores all paths through the program
- Great at catching hard to reproduce edge case bugs!

Static Analysis Improvements in Xcode 5

- Finds new kinds of issues
- Performs deeper code analysis
  - Objective-C
  - C++
- Exposes new workflows

Static Analysis Improvements in Xcode 5

- Finds new kinds of issues
- Performs deeper code analysis
  - Objective-C
  - C++
- Exposes new workflows

Collection API

```
newD = [NSDictionary dictionaryWithObject: defaultObj
                                   forKey: @”DefaultValue”];
```

dictionaryWithObject:forKey:

Creates and returns a dictionary containing a given key and value.

```
+ (id)dictionaryWithObject:(id)anObject forKey:(id < NSCopying >)aKey
```

Parameters

anObject

The value corresponding to *aKey*.

If this value is `nil`, an `NSInvalidArgumentException` is raised.

aKey

The key for *anObject*.

If this value is `nil`, an `NSInvalidArgumentException` is raised.

Return Value

A new dictionary containing a single object, *anObject*, for a single key, *aKey*.

dictionaryWithObject:forKey:

Creates and returns a dictionary containing a given key and value.

```
+ (id)dictionaryWithObject:(id)anObject forKey:(id < NSCopying >)aKey
```

Parameters

anObject

The value corresponding to *aKey*.

If this value is `nil`, an `NSInvalidArgumentException` is raised.

aKey

The key for *anObject*.

If this value is `nil`, an `NSInvalidArgumentException` is raised.

Return Value

A new dictionary containing a single object, *anObject*, for a single key, *aKey*.

Collection API Warnings

```
newD = [NSDictionary dictionaryWithObject: defaultObj
                                   forKey: @”DefaultValue”];
```

Collection API Warnings

```
defaultObj = nil;

newD = [NSDictionary dictionaryWithObject: defaultObj
                                   forKey: @”DefaultValue”];
```

Collection API Warnings

```
defaultObj = nil;

newD = [NSDictionary dictionaryWithObject: defaultObj
```

Value argument to 'dictionaryWithObject:forKey:' cannot be nil

Collection API Warnings

```
defaultObj = nil;

newD = @{ @"DefaultValue" : defaultObj };
```

Dictionary value cannot be nil

Collection API Warnings

```
- (void)listObjectAttributesChanged:(AIListObject *)inObject
                        modifiedKeys:(NSSet *)inModifiedKeys {

  id DNCenter = [NSNotificationCenter defaultCenter];
  BOOL shouldDelay = [self shouldDelayUpdates];

  [DNCenter postNotificationName:AttributesChanged
                          object:inObject
                        userInfo:(inModifiedKeys ?
                                  @{@"Keys" : inModifiedKeys} : nil)];


  if (!shouldDelay)
    [DNCenter postNotificationName:AttributeChangesComplete
                            object:inObject
                          userInfo:@{@"Keys" : inModifiedKeys}];    Dictionary value cannot be nil
}
```

Collection API Warnings

1. Assuming 'inModifiedKeys' is nil

Done

```
- (void)listObjectAttributesChanged:(AIListObject *)inObject
                        modifiedKeys:(NSSet *)inModifiedKeys {

  id DNCenter = [NSNotificationCenter defaultCenter];
  BOOL shouldDelay = [self shouldDelayUpdates];

  [DNCenter postNotificationName:AttributesChanged
                          object:inObject
                        userInfo:(inModifiedKeys ?                    1. Assuming 'inModifiedKeys' is nil
                                  @{@"Keys" : inModifiedKeys} : nil)];


  if (!shouldDelay)                                                   2. Assuming 'shouldDelay' is 0
    [DNCenter postNotificationName:AttributeChangesComplete
                            object:inObject
                          userInfo:@{@"Keys" : inModifiedKeys}];      3. Dictionary value cannot be nil
}
```

Collection API Warnings

2. Assuming 'shouldDelay' is 0 Done

```
- (void)listObjectAttributesChanged:(AIListObject *)inObject
                          modifiedKeys:(NSSet *)inModifiedKeys {

  id DNCenter = [NSNotificationCenter defaultCenter];
  BOOL shouldDelay = [self shouldDelayUpdates];

  [DNCenter postNotificationName:AttributesChanged
                          object:inObject
                        userInfo:(inModifiedKeys ?                    1. Assuming 'inModifiedKeys' is nil
                                  @{@"Keys" : inModifiedKeys} : nil)];

  if (!shouldDelay)                                                   2. Assuming 'shouldDelay' is 0
    [DNCenter postNotificationName:AttributeChangesComplete
                          object:inObject
                        userInfo:@{@"Keys" : inModifiedKeys}];        3. Dictionary value cannot be nil
}
```

Collection API Warnings

3. Dictionary value cannot be nil

Done

```
- (void)listObjectAttributesChanged:(AIListObject *)inObject
                        modifiedKeys:(NSSet *)inModifiedKeys {

  id DNCenter = [NSNotificationCenter defaultCenter];
  BOOL shouldDelay = [self shouldDelayUpdates];

  [DNCenter postNotificationName:AttributesChanged
                          object:inObject
                        userInfo:(inModifiedKeys ?                    1. Assuming 'inModifiedKeys' is nil
                                  @{@"Keys" : inModifiedKeys} : nil)];


  if (!shouldDelay)                                                   2. Assuming 'shouldDelay' is 0
    [DNCenter postNotificationName:AttributeChangesComplete
                          object:inObject
                        userInfo:@{@"Keys" : inModifiedKeys}];        3. Dictionary value cannot be nil
}
```

C++ Mismatched Allocators

```
Motion *throwMotion = new Motion("throw");
useMotion(throwMotion);
free(throwMotion);
```

C++ Mismatched Allocators

```
Motion *throwMotion = new Motion("throw");
us
fr
```

```
Motion *throwMotion = new Motion("throw");        1. Memory is allocated

useMotion(throwMotion);

free(throwMotion);        2. Memory allocated by 'new' should be deallocated by 'delete', not free()
```

New Analyzer Warnings

- Adding nil to NSMutableArray
- Using nil as key or value for NSMutableDictionary
- C++ use-after-free
- C++ mismatched deallocators
- C++ creation of references to null

Growing Body of Issues Found (WWDC 2012)

- Violation of reference counting rules
- Improper memory management (malloc & free)
- Misuse of Grand Central Dispatch API
- Null dereference
- Use of non-secure Unix APIs
- Violation of 'self = [super init]' rule
- @synchronized with nil mutex
- Dead stores
- Use of uninitialized values

Static Analysis Improvements in Xcode 5

- Finds new kinds of issues
- Performs deeper code analysis
  - Objective-C
  - C++
- Exposes new workflows

Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                       inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                     inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```

Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                       inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                     inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```

Objective-C Cross Method Analysis

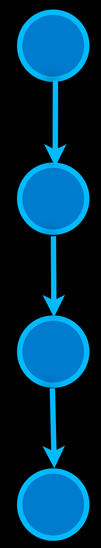
```
path = [Asset pathForAssetName:name
                      inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                    inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```

Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                        inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                      inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```

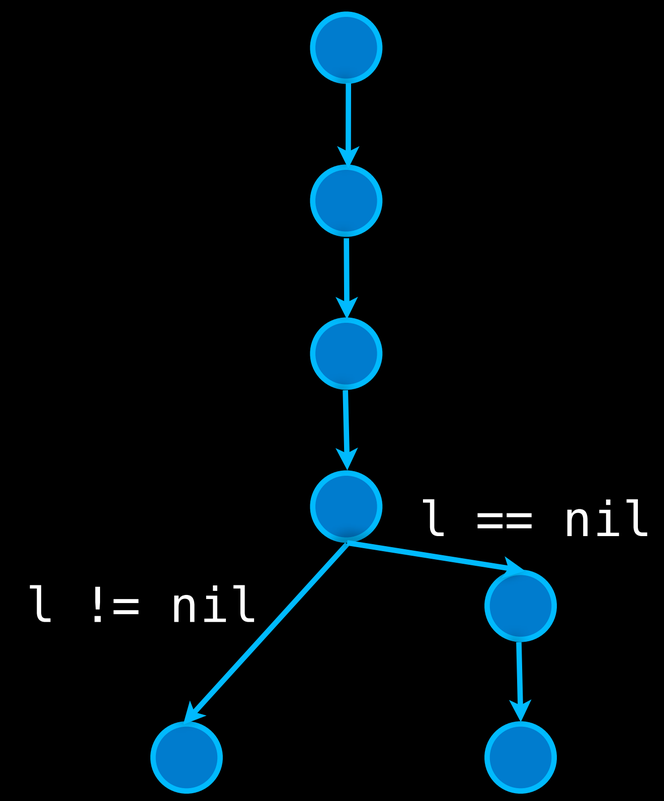
Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                       inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                     inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```



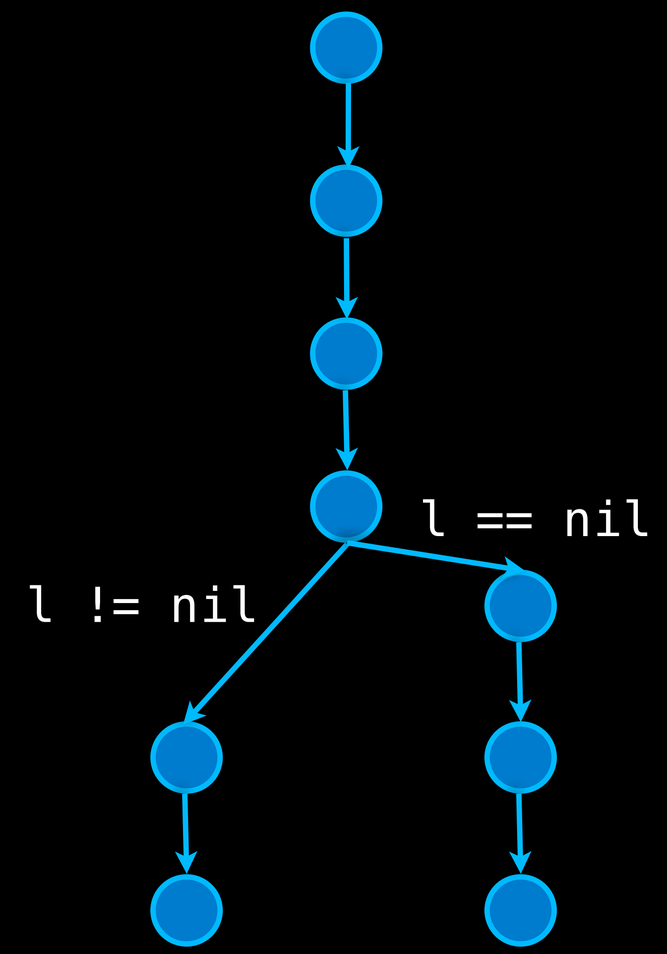
Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                       inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                     inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```



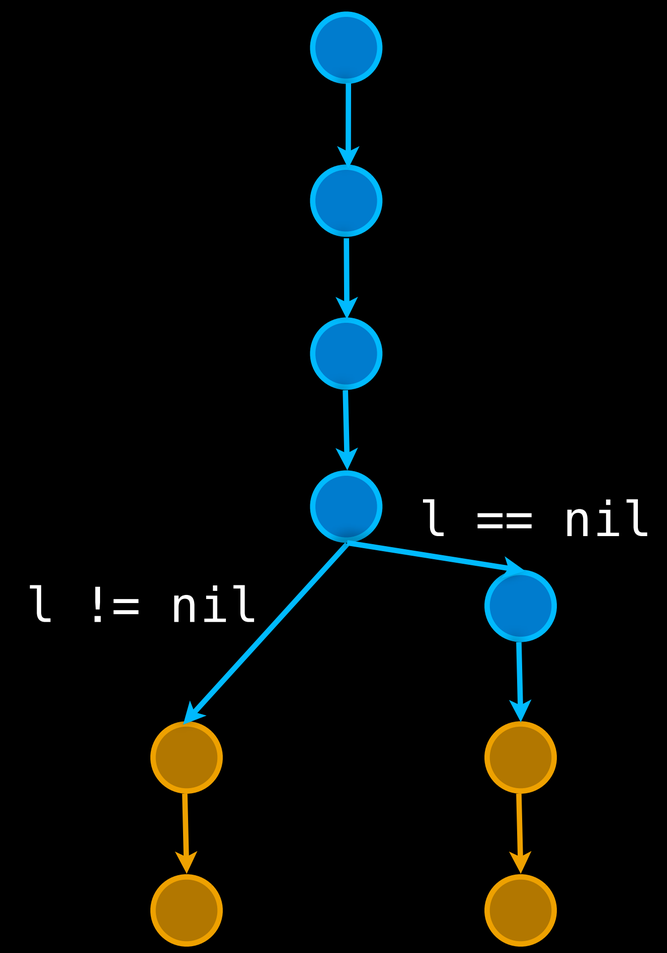
Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                       inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                     inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```



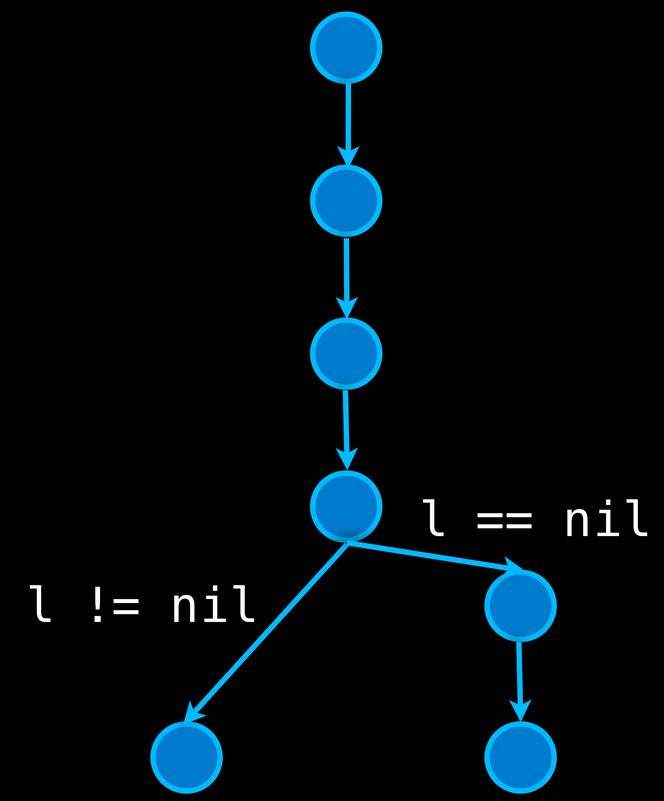
Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                       inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                     inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```



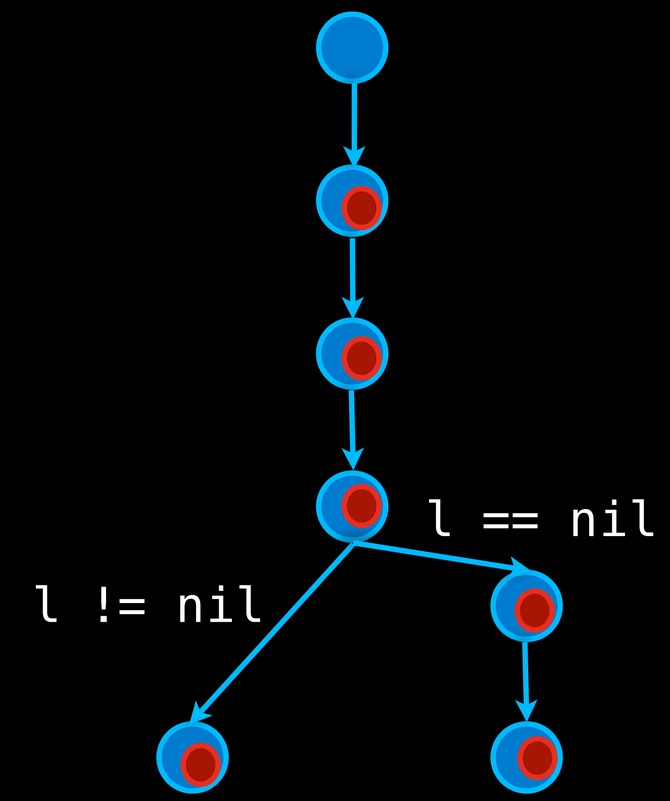
Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                       inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                     inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```



Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                        inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                      inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```

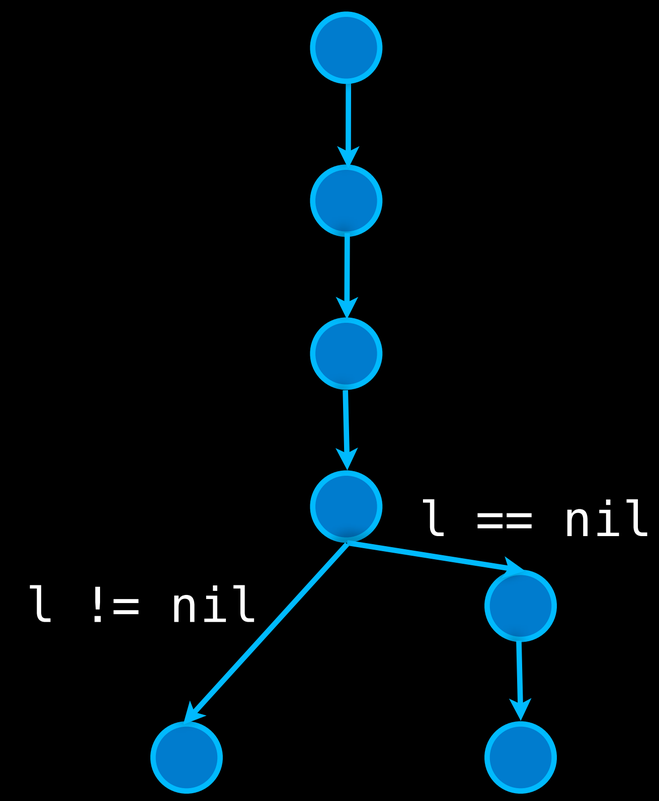


- a is of type `Asset`

Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                      inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                    inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```

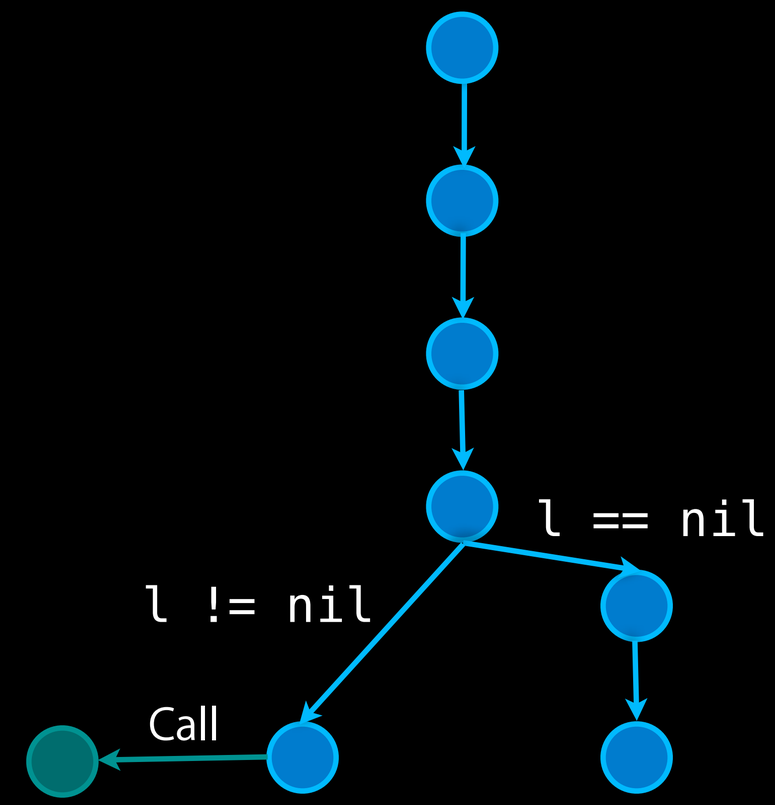
```
- (void)setName:(NSString *)name
      localized:(NSString *)l {
  _properties[@"localizedName"] = l;
  _properties[@"name"] = name;
}
```



Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                      inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                    inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```

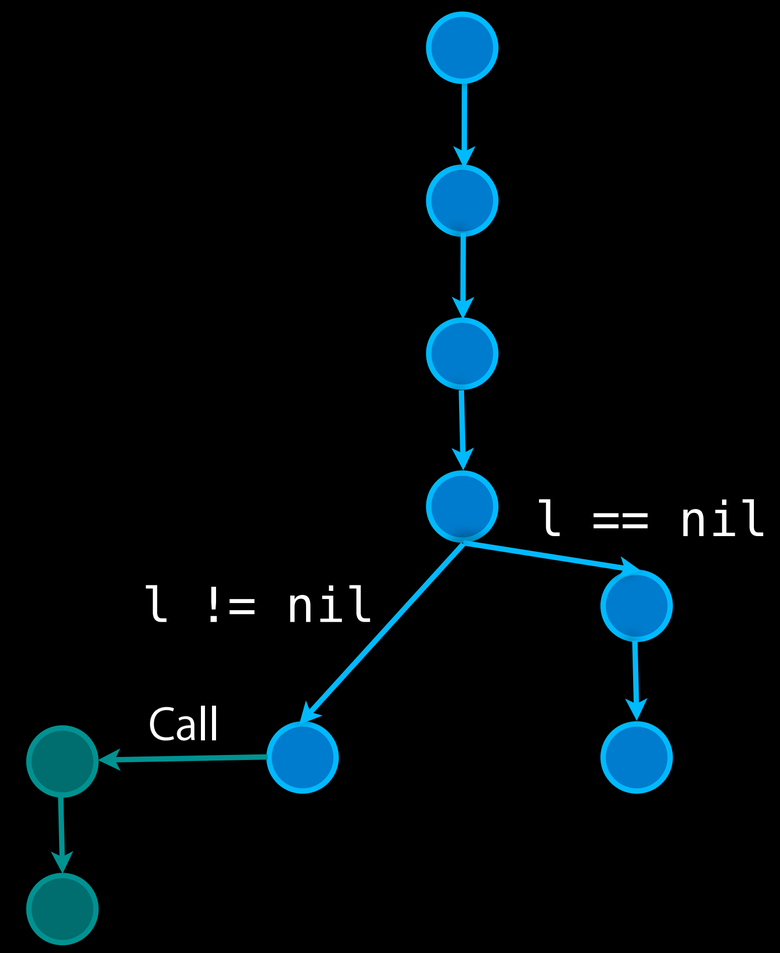
```
- (void)setName:(NSString *)name
      localized:(NSString *)l {
  _properties[@"localizedName"] = l;
  _properties[@"name"] = name;
}
```



Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                      inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                    inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```

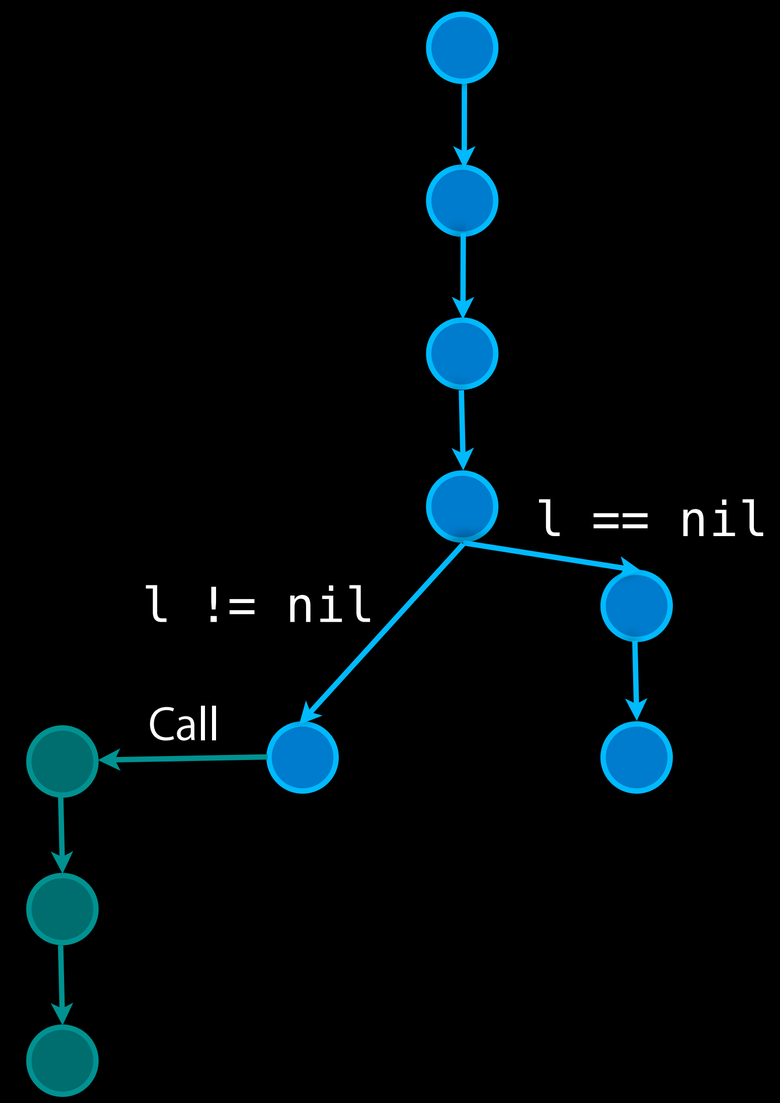
```
- (void)setName:(NSString *)name
      localized:(NSString *)l {
  _properties[@"localizedName"] = l;
  _properties[@"name"] = name;
}
```



Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                      inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                    inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```

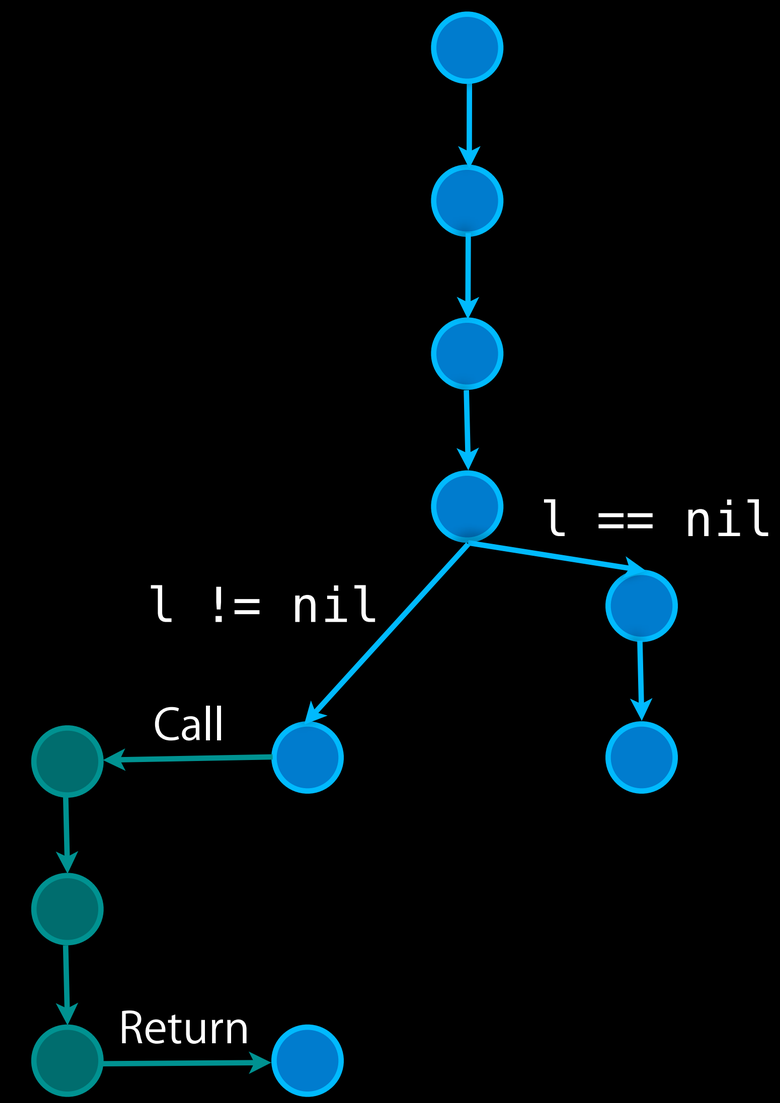
```
- (void)setName:(NSString *)name
      localized:(NSString *)l {
  _properties[@"localizedName"] = l;
  _properties[@"name"] = name;
}
```



Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                      inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                    inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```

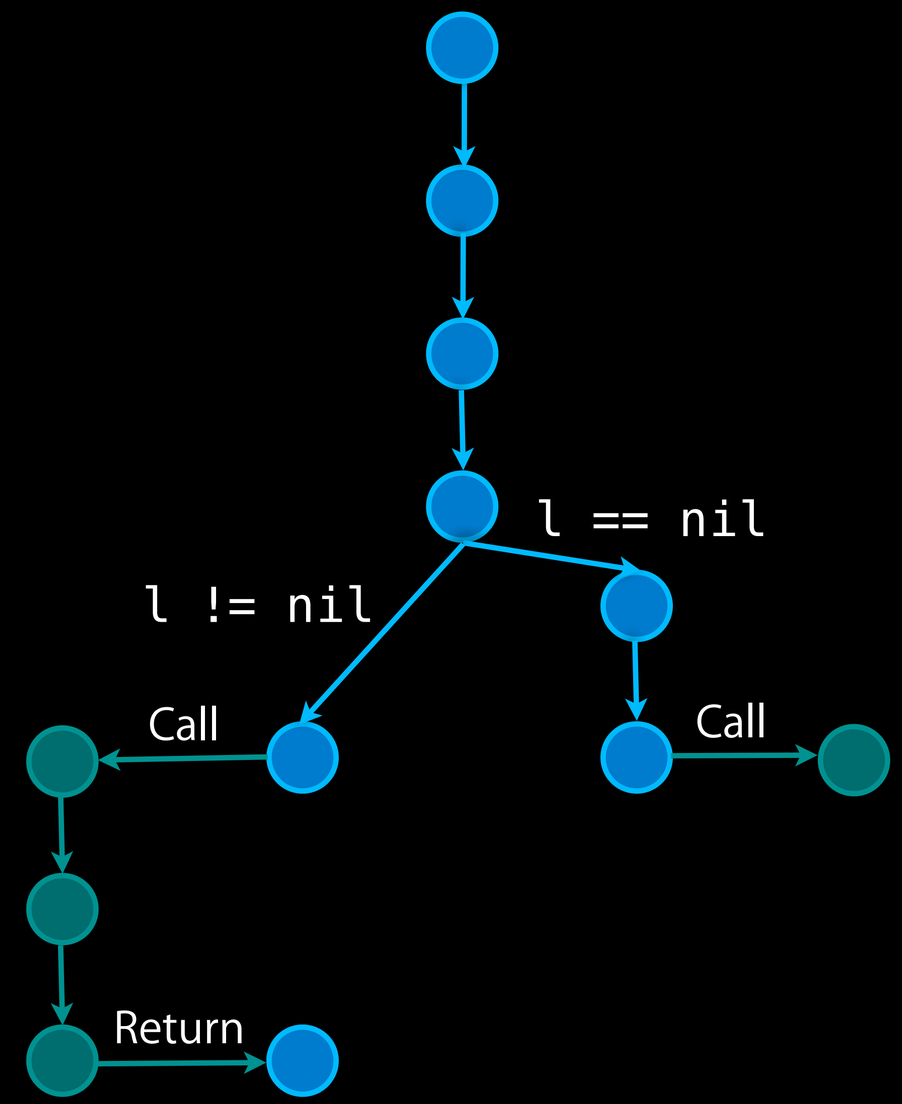
```
- (void)setName:(NSString *)name
      localized:(NSString *)l {
  _properties[@"localizedName"] = l;
  _properties[@"name"] = name;
}
```



Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                      inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                    inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```

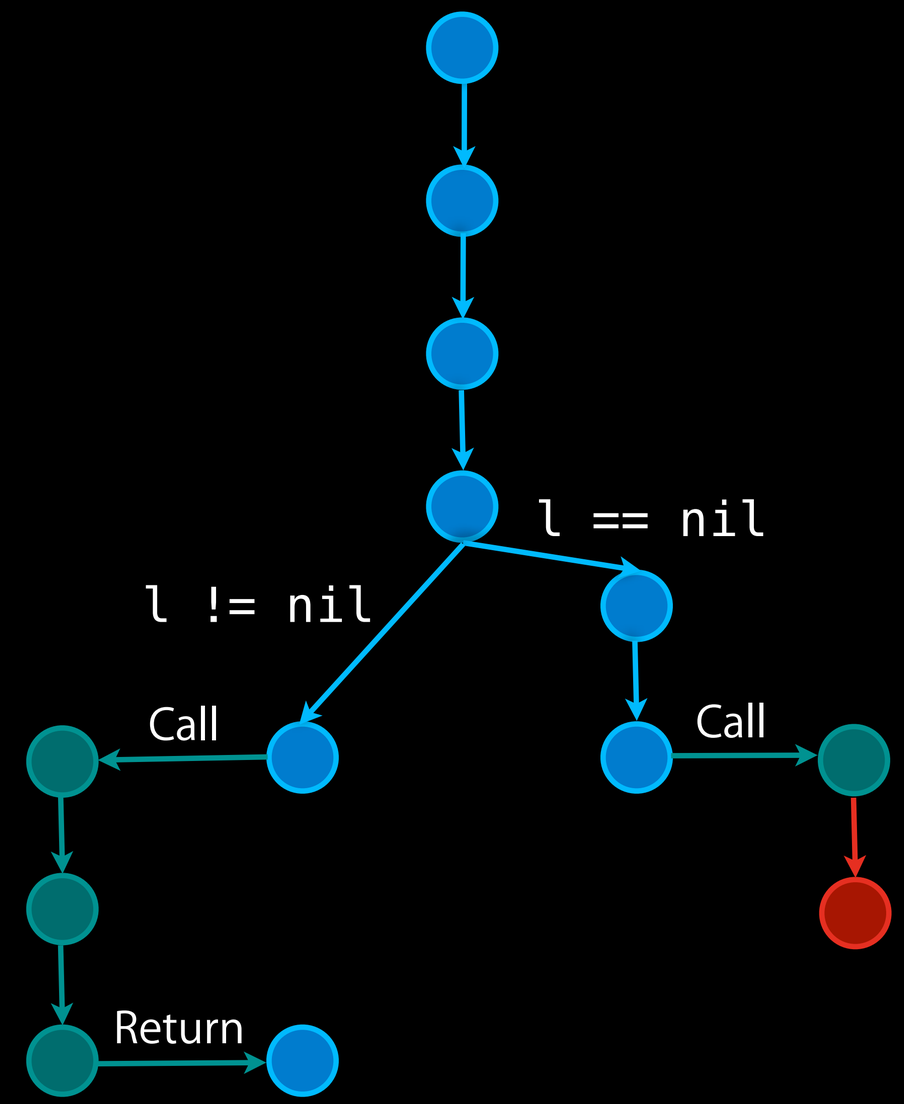
```
- (void)setName:(NSString *)name
      localized:(NSString *)l {
  _properties[@"localizedName"] = l;
  _properties[@"name"] = name;
}
```



Objective-C Cross Method Analysis

```
path = [Asset pathForAssetName:name
                      inBundle:bundle];
id a = [[Asset alloc] initWithPath:path];

l = [self localizedAssetName:name
                    inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

[a setName:name localized:l];
```

```
- (void)setName:(NSString *)name
      localized:(NSString *)l {
  _properties[@"localizedName"] = l;
  _properties[@"name"] = name;
}
```



Objective-C Cross Method Analysis

```
+ (Asset*)localizedAssetWithName:(NSString *)name
                         inBundle:(NSBundle *)bundle {
  NSString *path;
  NSString *l;

  path = [Asset pathForAssetName:name
                         inBundle:bundle];
  id a = [[Asset alloc] initWithPath:path];

  l = [self localizedAssetName:name                    1. Value assigned to 'l'
                         inBundle:bundle];
  if (!l)                                              2. Assuming 'l' is nil
    NSLog(@"unable to localize '%@'", name);

  [a setName:name localized:l];                        3. Passing nil object reference via 2nd parameter 'l'
  return a;                                            4. Calling 'setName:localized:'
}

- (void)setName:(NSString *)name                       5. Entered call from 'localizedAssetWithName:inBundle:'
      localized:(NSString *)l {
  _properties[@"localizedName"] = l;                   6. Value stored into 'NSMutableDictionary' cannot be nil
  _properties[@"name"] = name;
}
```

C++ Cross Method Analysis

```
String String::getAsText() {
  String Result;
  unsigned OutCount;

  auto_ptr<UniChar> Buffer(new UniChar[uniCharBufferSize]);
  convertFromUnicodeToText(uniCharBufferSize * sizeof(UniChar),
                           &OutCount,
                           Buffer.get());
  Result.Set(Buffer.get(), OutCount / sizeof(UniChar));

  return Result;
}
```

C++ Cross Method Analysis

```
String String::getAsText() {
  String Result;
  unsigned OutCount;

  auto_ptr<UniChar> Buffer(new UniChar[uniCharBufferSize]);
  convertFromUnicodeToText(uniCharBufferSize * sizeof(UniChar),
                           &OutCount,
                           Buffer.get());
  Result.Set(Buffer.get(), OutCount / sizeof(UniChar));

  return Result;   Memory allocated by 'new[]' should be deallocated by 'delete[]', not 'delete' (within a call to '~auto_ptr')
}
```

C++ Cross Method Analysis

```
String String::getAsText() {
  String Result;
  unsigned OutCount;

  auto_ptr<UniChar> Buffer(new UniChar[uniCharBufferSize]);          1. Memory is allocated
  convertFromUnicodeToText(uniCharBufferSize * sizeof(UniChar),
                           &OutCount,
                           Buffer.get());
  Result.Set(Buffer.get(), OutCount / sizeof(UniChar));

  return Result;                                                     2. Calling '~auto_ptr'
}
```

C++ Cross Method Analysis

```
String String::getAsText() {
  String Result;
  unsigned OutCount;

  auto_ptr<UniChar> Buffer(new UniChar[uniCharBufferSize]);          1. Memory is allocated
  convertFromUnicodeToText(uniCharBufferSize * sizeof(UniChar),
                           &OutCount,
                           Buffer.get());
  Result.Set(Buffer.get(), OutCount / sizeof(UniChar));

  return Result;                                                     2. Calling '~auto_ptr'
}
```

```
    _LIBCPP_INLINE_VISIBILITY ~auto_ptr() throw() {delete __ptr_;}
                                                  3. Entered call from 'String::getAsText'
    LIBCPP_INLINE_VISIBILITY                      4. Memory allocated by 'new[]' should be deallocated by 'delete[]', not 'delete'
```

C++ Cross Method Analysis

```
String String::getAsText() {
  String Result;
  unsigned OutCount;

  auto_ptr<UniChar> Buffer(new UniChar[uniCharBufferSize]);          1. Memory is allocated
  convertFromUnicodeToText(uniCharBufferSize * sizeof(UniChar),
                           &OutCount,
                           Buffer.get());
  Result.Set(Buffer.get(), OutCount / sizeof(UniChar));

  return Result;                                                     2. Calling '~auto_ptr'
}
```

```
    _LIBCPP_INLINE_VISIBILITY ~auto_ptr() throw() {delete __ptr_;}
                                3. Entered call from 'String::getAsText'
    LIBCPP_INLINE_VISIBILITY    4. Memory allocated by 'new[]' should be deallocated by 'delete[]', not 'delete'
```

Great for C++ since so many implementation details are in the headers!

What Made it Possible?

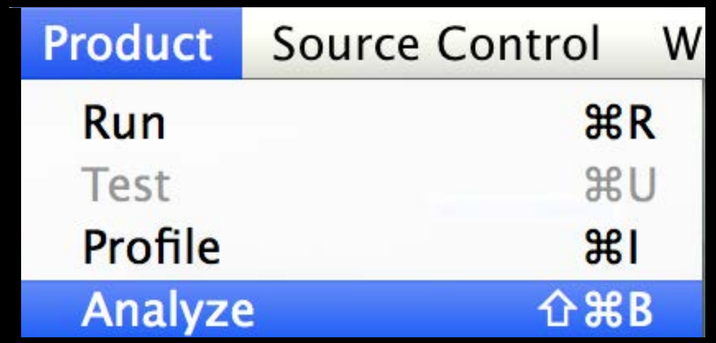
- Path-specific tracking of object types
- Constructors and destructors are visible (C++)
- All methods from a source file and included headers are visible
- Function implementations from other source files are not visible

Static Analysis Improvements in Xcode 5

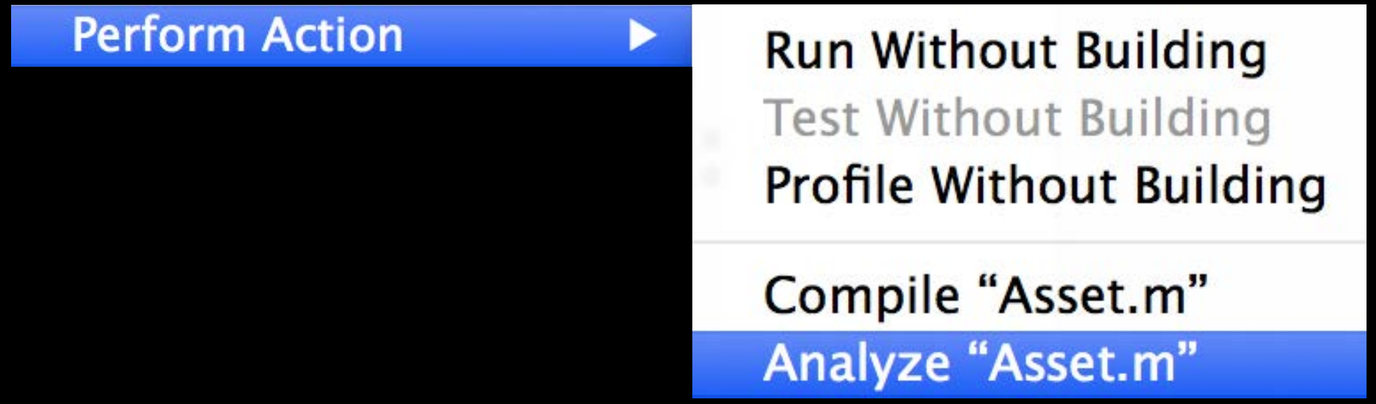
- Finds new kinds of issues
- Performs deeper code analysis
  - Objective-C
  - C++
- Exposes new workflows

How to Run the Analyzer?

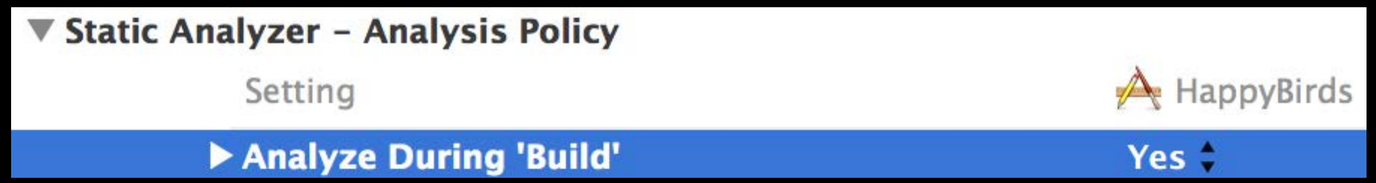
- Analyze on demand



- Analyze a single file



- Analyze during Build



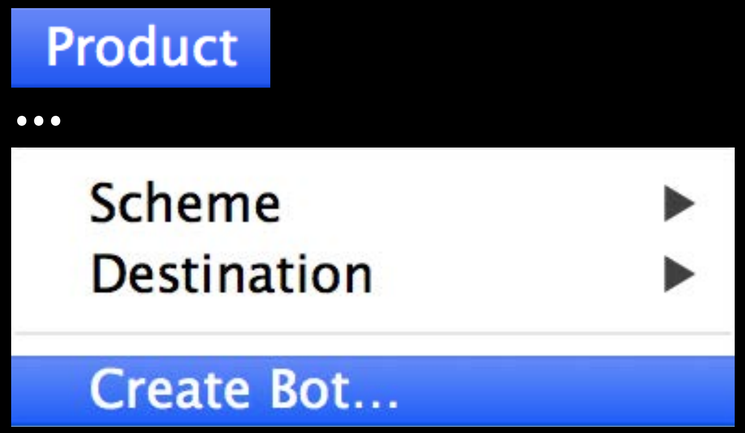
Continuous Integration

- For projects free of analyzer warnings

Run from command line (the same as Product → Analyze)

```
$ xcodebuild analyze -project HappyBirds.xcodeproj
```

- Set up a Bot that runs the analyzer



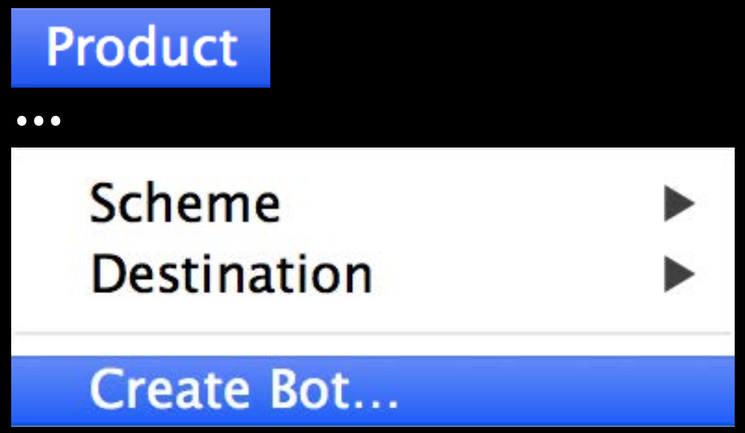
Continuous Integration

- For projects free of analyzer warnings

Run from command line (the same as Product → Analyze)

```
$ xcodebuild analyze -project HappyBirds.xcodeproj
```

- Set up a Bot that runs the analyzer



Trading Off Analysis Power

Shallow ⟷ **Deep**

Quick analysis — More thorough analysis

Static Analyzer – Analysis Policy

Setting	HappyBirds
Analyze During 'Build'	**Yes**
Mode of Analysis for 'Analyze'	Deep
Mode of Analysis for 'Build'	Shallow (faster)

Trading Off Analysis Power

Shallow ⟷ **Deep**

Quick analysis — More thorough analysis

▼ **Static Analyzer – Analysis Policy**

Setting	HappyBirds
Analyze During 'Build'	**Yes**
Mode of Analysis for 'Analyze'	Deep
Mode of Analysis for 'Build'	Shallow (faster)

Always analyze in Deep mode as part of qualifications!

Getting More out of Your Comments

Where is the Documentation?

```
@interface Asset : NSObject

@property NSMutableDictionary *properties;

+ (NSString *)pathForAssetName:(NSString *)name
                      inBundle:(NSBundle *)bundle;
@end
```

Structured Comments (Doxygen)

```
/// My Asset class.
@interface Asset : NSObject

/// A container for \c Asset properties.
@property NSMutableDictionary *properties;

/// \brief Locates an asset within a bundle.
///
/// \param name The name of the asset.
/// \param bundle The bundle in which the asset is stored.
/// \returns The path to the asset.
+ (NSString *)pathForAssetName:(NSString *)name
                      inBundle:(NSBundle *)bundle;
@end
```

Structured Comments (Doxygen)

```
/// My Asset class.
@interface Asset : NSObject

/// A container for \c Asset properties.
@property NSMutableDictionary *properties;

/// \brief Locates an asset within a bundle.
///
/// \param name The name of the asset.
/// \param bundle The bundle in which the asset is stored.
/// \returns The path to the asset.
+ (NSString *)pathForAssetName:(NSString *)name
                      inBundle:(NSBundle *)bundle;
@end
```

Structured Comments (Doxygen)

```
/// My Asset class.
@interface Asset : NSObject

/// A container for \c Asset properties.
@property NSMutableDictionary *properties;

/// \brief Locates an asset within a bundle.
///
/// \param name The name of the asset.
/// \param bundle The bundle in which the asset is stored.
/// \returns The path to the asset.
+ (NSString *)pathForAssetName:(NSString *)name
                     inBundle:(NSBundle *)bundle;
@end
```

Quick Help from the Editor

```
path = [Asset pathForAssetName:name
                      inBundle:bundle];
```

Declared In Asset.h

```
l = [self localizedAssetName:name
                    inBundle:bundle];
if (!l)
  NSLog(@"unable to localize '%@'", name);

  [a setName:name localized:l];
  return a;
}
```

Option + Click

Quick Help from the Editor

```
path = [Asset pathForAssetName:name
                      inBundle:bundle];
```

Declaration	`+ (NSString *)pathForAssetName:(NSString *)name inBundle: (NSBundle *)bundle;`
Description	Locates an asset within a bundle.
Parameters	name — The name of the asset.
	bundle — The bundle in which the asset is stored.
Returns	The path to the asset.
Declared In	Asset.h

Option + Click

Quick Help Inspector

HappyBirds.xcodeproj — Asset.m

HappyBirds 〉 My Mac 64-bit | Clean HappyBirds: **Succeeded** | Today at 10:33 AM | No Issues

Asset.m

Happ... 〉 Asset.m 〉 +localizedAssetWithName:inBundle:

HappyBirds
2 targets, OS X SDK 10.9

- HappyBirds
- HappyBirdsTests
- Frameworks
- Products

```
+ (Asset*)localizedAssetWithName:(NSString *)name
                        inBundle:(NSBundle *)bundle {
  NSString *path;
  NSString *l;

  path = [Asset pathForAssetName:name
                        inBundle:bundle];
  id a = [[Asset alloc] initWithPath:path];

  l = [self localizedAssetName:name
                       inBundle:bundle];
  if (!l)
    NSLog(@"unable to localize '%@'", name);

  [a setName:name localized:l];
  return a;
}
```

Quick Help

Declaration + (NSString *)pathForAssetName:(NSString *)name inBundle: inBundle:(NSBundle *)bundle;

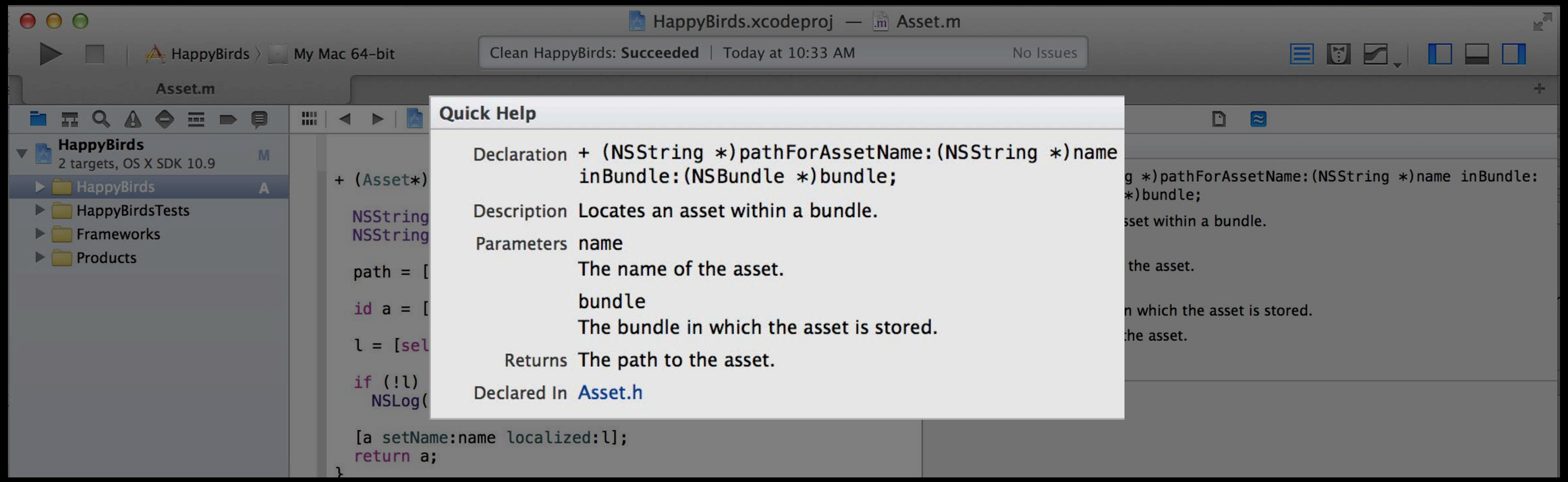
Description Locates an asset within a bundle.

Parameters name
The name of the asset.
bundle
The bundle in which the asset is stored.

Returns The path to the asset.

Declared In Asset.h

Quick Help Inspector



Documentation in Code Completion

```
path = [Asset pathForAssetName:(NSString *) inBundle:(NSBundle *)
M  NSString * pathForAssetName:(NSString *) inBundle:(NSBundle *)
```

Locates an asset within a bundle.

How it Works

- Compiler processes the comments along with user code
- Attaches the comments to declarations

```
/// A container for \c Asset properties.
@property NSMutableDictionary *properties;

/// \brief Locates an asset within a bundle.
///
/// \param name The name of the asset.
/// \param bundle The bundle in which the asset is stored.
/// \return The path to the asset.
+ (NSString *)pathForAssetName:(NSString *)name
                      inBundle:(NSBundle *)bundle;
```

How it Works

- Compiler processes the comments along with user code
- Attaches the comments to declarations

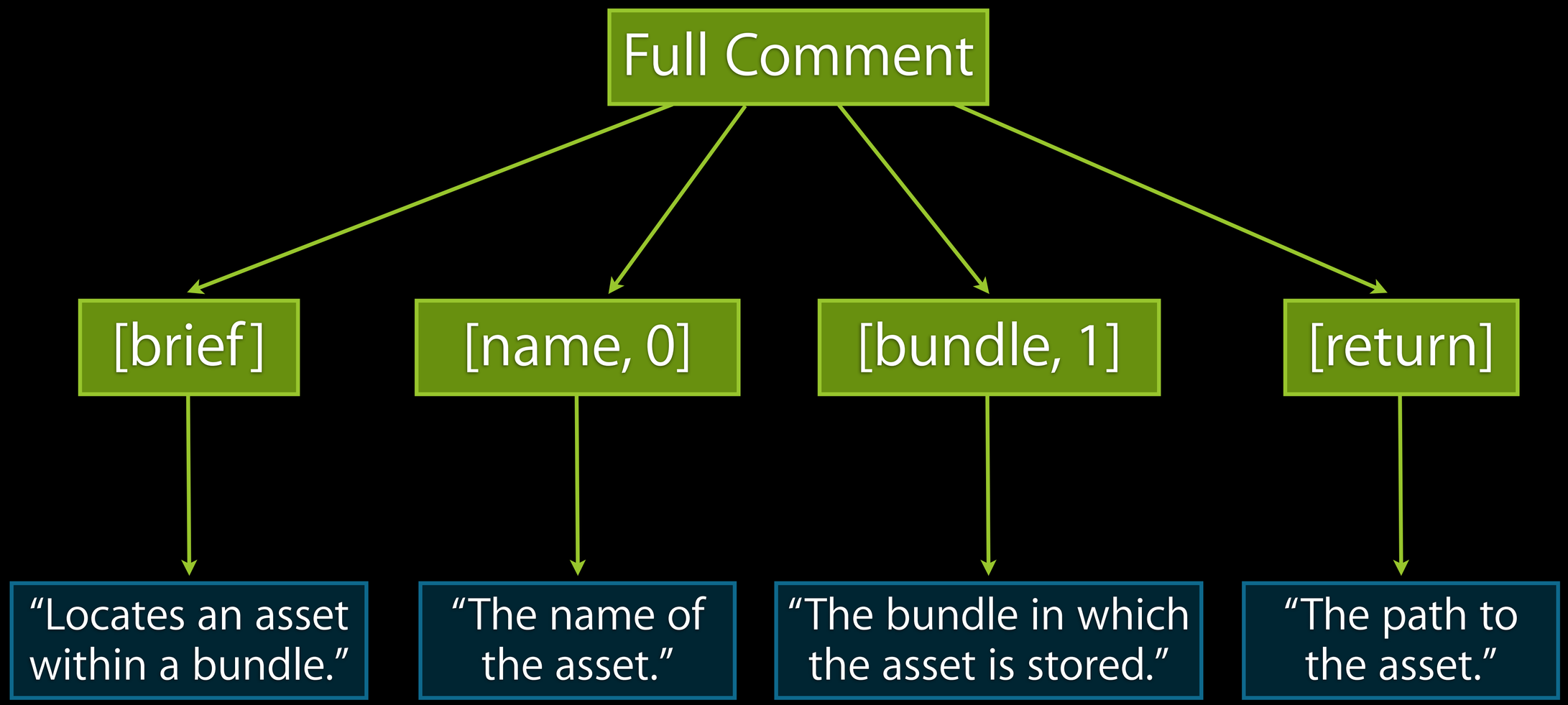
```
/// A container for \c Asset properties.
@property NSMutableDictionary *properties;

/// \brief Locates an asset within a bundle.
///
/// \param name The name of the asset.
/// \param bundle The bundle in which the asset is stored.
/// \return The path to the asset.
+ (NSString *)pathForAssetName:(NSString *)name
                      inBundle:(NSBundle *)bundle;
```

How it Works

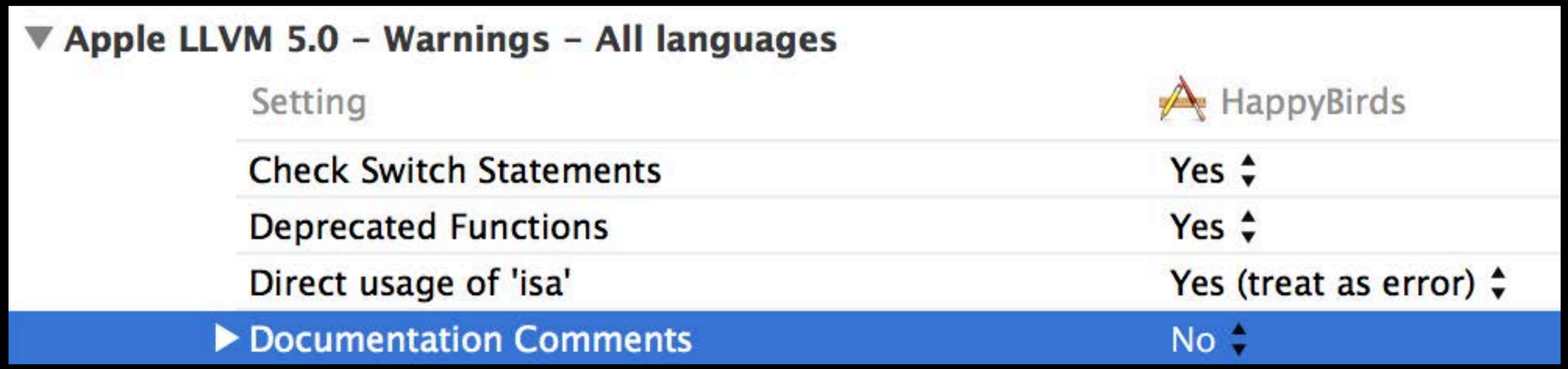
- Compiler processes the comments along with user code
- Attaches the comments to declarations

```
/// A container for \c Asset properties.
@property NSMutableDictionary *properties;

/// \brief Locates an asset within a bundle.
///
/// \param name The name of the asset.
/// \param bundle The bundle in which the asset is stored.
/// \return The path to the asset.
+ (NSString *)pathForAssetName:(NSString *)name
                      inBundle:(NSBundle *)bundle;
```

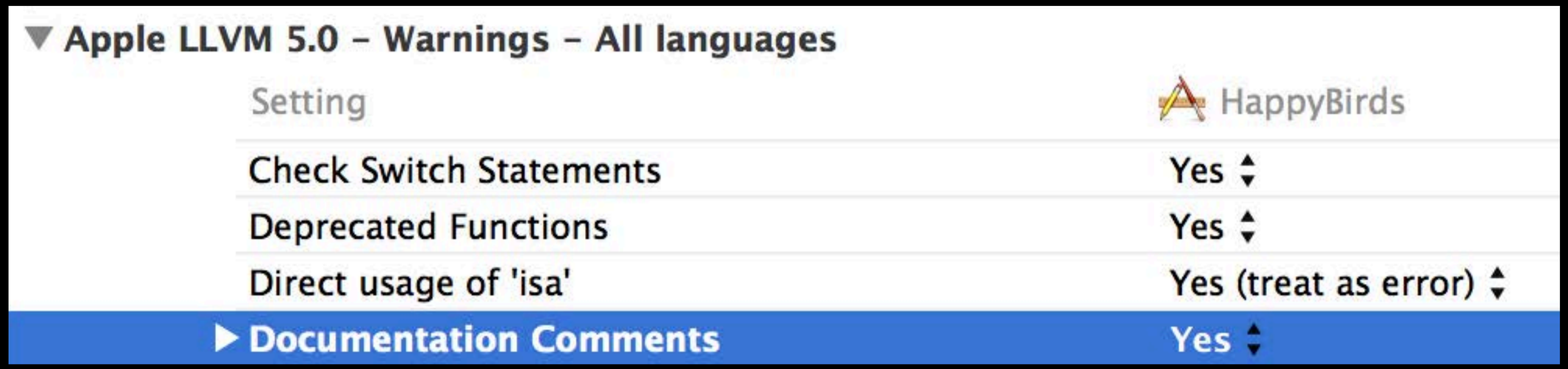
Precise Representation of Comments



Diagnostics (-Wdocumentation)



Diagnostics (-Wdocumentation)



-Wdocumentation

```
/// \brief Sets the name and the localized name.
///
/// \param name The name of the asset.
/// \param l The localized name.
/// \returns error code.

- (void)setName:(NSString *)name localized:(NSString *)l;
```

```
warning: '\returns' command used in a comment that is attached to a
         method returning void
/// \returns error code.
    ~^~~~~~~~~~~~~~~~~~~
```

-Wdocumentation

```
/// \brief
/// \param name The name of the asset.
/// \param bundle The bundle in which the asset is stored.
/// \returns The path to the asset.

+ (NSString *)pathForAssetName:(NSString *)name
                      inBundle:(NSBundle *)bundle;
```

```
warning: empty paragraph passed to '\brief' command
/// \brief
    ~~~~~~^
```

Typo Correction

```
/// \brief Locates an asset within a bundle.
///
/// \param name The name of the asset.
/// \param bungle The bundle in which the asset is stored.
/// \returns The path to the asset.
+ (NSString *)pathForAssetName:(NSString *)name
                      inBundle:(NSBundle *)bundle;
```

Parameter 'bungle' not found in the function declaration

Typo Correction

Issue Parameter 'bungle' not found in the function declaration

Fix-it Did you mean 'bundle'?

```
a bundle.
/// \param ame The name of the asset.
/// \param bundle The bundle in which the asset is stored.
/// \returns The path to the asset.
+ (NSString *)pathForAssetName:(NSString *)name
                       inBundle:(NSBundle *)bundle;
```

Parameter 'bungle' not found in the function declaration

Typo Correction

```
/// \brief Locates an asset within a bundle.
///
/// \param name The name of the asset.
/// \param bundle The bundle in which the asset is stored.
/// \returns The path to the asset.
+ (NSString *)pathForAssetName:(NSString *)name
                      inBundle:(NSBundle *)bundle;
```

All Wired Up—Ready for Your Comments!

- Write the comments
  - Doxygen
  - HeaderDoc
- See your comments in Quick Help and code completion
- Turn on Documentation Comments warning

http://www.doxygen.org
http://developer.apple.com/library/mac/#documentation/DeveloperTools/Conceptua;/Headerdoc

Summary

- Produces faster apps
  - Support for latest hardware
  - Aggressive new optimizations
- Streamlines developer experience
  - Easier tools installation
  - C++11 support
  - Stricter warnings
  - Deeper static analysis
  - Documentation in comments

Apple LLVM 5.0

More Information

Dave DeLong
Developer Tools Evangelist
delong@apple.com

LLVM Project
Open-Source LLVM Project Home
http://llvm.org

Clang Static Analyzer
Open-Source Clang Static Analyzer
http://clang-analyzer.llvm.org

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Optimize Your Code Using LLVM	Nob Hill Wednesday 3:15PM
Advances in Objective-C	Mission Tuesday 4:30PM
Continuous Integration with Xcode 5	Presidio Tuesday 3:15PM

Labs

LLVM	Tools Lab C Tuesday 3:15PM
Objective-C & LLVM	Tools Lab B Wednesday 9:00AM
Objective-C & LLVM	Tools Lab C Thursday 2:00 PM

WWDC2013