# Objective-C

# Wo kommt Objective-C zum Einsatz

OSX mit Cocoa

iOS mit CocoaTouch

# Was steckt hinter Objective-C?

Smalltalk
1972

Objective-C
frühen 80er
von Brad Cox & Tome Love

NeXTSTEP
1989

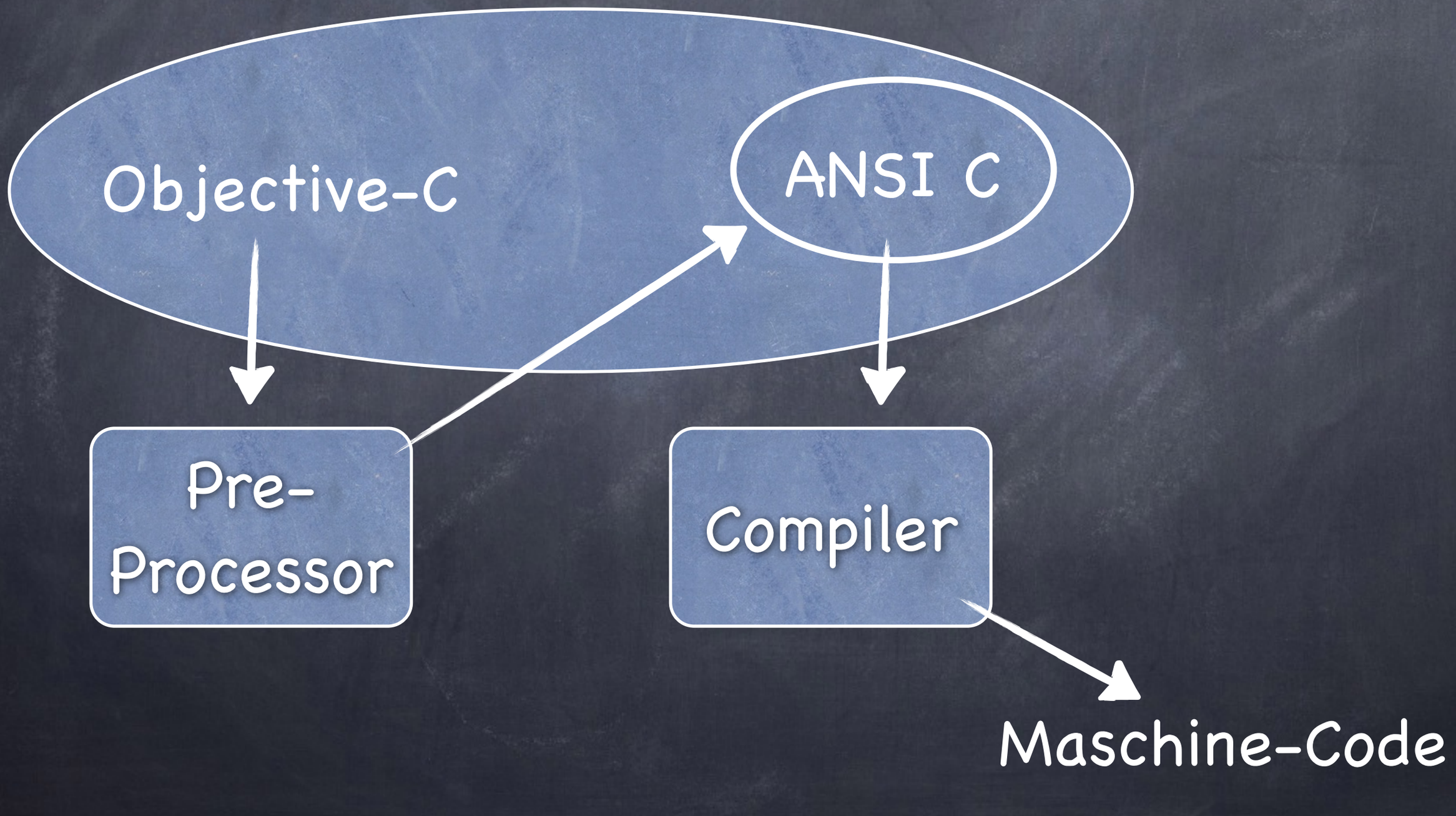
1996 gekauft

OS X
2001

iOS
2007

C
1972

# Wie sieht die Sprache heute aus?



# Tutorial: Cold Water!

## Customer.h

```objc
#import <Foundation/Foundation.h>



@interface Customer : NSObject



@end
```

## Customer.m

```objc
#import "Customer.h"

@interface Customer ()

@end

@implementation Customer



@end
```

## Customer.h

```
@import Foundation;




@interface Customer : NSObject

@property (nonatomic, strong)
NSString* firstName;




@end
```

## Customer.m

```
#import "Customer.h"

@interface Customer ()

@end

@implementation Customer




@end
```

## Customer.h

```
@import Foundation;



@interface Customer : NSObject



@end
```

## Customer.m

```
#import "Customer.h"

@interface Customer ()

@end

@implementation Customer



@end
```

## Customer.h

```
@import Foundation;



@interface Customer : NSObject



@end
```

## Customer.m

```
#import "Customer.h"

@interface Customer ()

@end

@implementation Customer



@end
```

## Customer.h

```
@import Foundation;



@interface Customer : NSObject



@end
```

## Customer.m

```
#import "Customer.h"

@interface Customer ()

@end

@implementation Customer



@end
```

## Customer.h

```
@import Foundation;



@interface Customer : NSObject



@end
```

## Customer.m

```
#import "Customer.h"

@interface Customer ()

@end

@implementation Customer



@end
```

# Erste Klasse erstellen

# Erste Variable

# Erstes Property

# Erste Methode

# Tutorial: Hello World!

# Erste eigene Instanz

# Tutorial: Array

# Sprach-Merkmale

- OO-Sprache mit Verwandtschaft zu Smalltalk
- Struktur sowie Kontrollanweisungen ähnlich zu C
- Unterscheidung zwischen Objekten und Strukturen (Pointer!)
- Kein Garbage Collector
- Exception Handling ähnlich Java oder C#
- Dynamisch gebundene und **dynamisch typisierte** Sprache

# Literale

- NSString `@”Hello”`
- NSNumber `@1234`
- NSArray `@[@”Jan”, @”Feb”, @”Mrz”]`
- NSDictionary `@{@1 : @”Jan”, @12 : @”Dez”}`

# Protocols

Deklaration des Protokolls

Implementation des Protokolls

- Entspricht Interfaces in Java oder C#

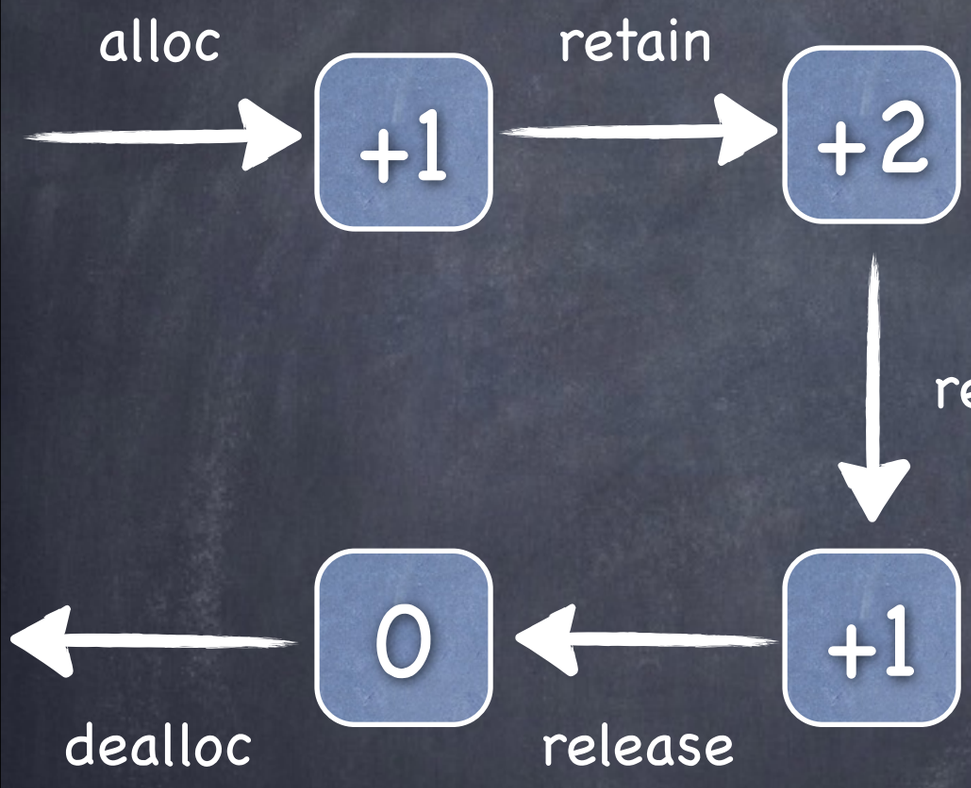
```
@protocol MyCommand {      // Command
- (void)execute:(ContextInformation*)context;
}

@interface MyConcreteCommand : NSObject<MyCommand>
@end
@implementation MyConcreteCommand // Co
- (void)execute:(ContextInformation*)co
@end

@implementation MyInvoker
- (void)doSomethin {
    id<MyCommand> command = [factory createCommand];
    [command execute:nil];
}
@end
```

Verwendung des Protokolls

# Memory Management

- Jedes Objekt hat einen Referenzzähler
- Nach dem `alloc`: +1
- Wenn der Referenzzähler auf 0 sinkt
  - Objekt wird automatisch freigegeben

# Memory Management II




```
NSMutableArray* array =
 [[NSMutableArray alloc]
                initWithCapacity:1];

User* user [[User alloc] init];  // +1

[array addObject:user];          // +2

          - (void)addObject:(id)object {
              [object retain];
              .....
          }

[user release];                  // +1

[array release];                 //  0
```

# Memory Management (ARC)

```
NSMutableArray* array =
 [[NSMutableArray alloc] initWithCapacity:1];

User* user [[User alloc] init];

[array addObject:user];
```

```
- (void)addObject:(id)object {
    [object retain];
    .....
}
```

```
[User release];

[array release];
```

# Tutorial: First Class