Macoun'10

Eleganter Code dank Blocks

Frank Illenberger
ProjectWizards

Was sind Blocks?

- Apple-eigene Erweiterung des C-Standards
- explizit entwickelt für Grand Central Dispatch
- anonyme Funktion
  - Argumente
  - interne Variablen
  - fängt externe Variablen ein (Closure, achtung retain-Cycles!)

Was sind Blocks?

- Code wird zu einer Art von Daten
  - Zuweisen zu Variablen
  - herumreichen als Funktionsparameter
- Ähnlich zu C-Funktionszeigern
  - gleicher Aufruf
  - Blocks werden jedoch zur Laufzeit erzeugt (Stack)

Alle Beispiele verwenden Garbage-Collection

iOS → bugreport.apple.com

Was sind Blocks?

```
typedef int (^CounterBlock)();

Counter newCounter(int start, int step) {
    __block int index = start;
    Counter counter = ^{ return (index += step); }
    return [counter copy];
}

int main(void) {
    Counter counter = newCounter(2, 3);
    NSLog("%d-%d-%d", counter(), counter(), counter());
    return 0;
}

/* Output:  2-5-7 */
```

Was sind Blocks?

- Ermöglichen Konstrukte aus dynamischen Sprachen wie Lisp, Python, JavaScript oder Ruby
- Synchrones Umschließen (Wrapping)
- Asynchroner Aufruf (Callback, Delegation)

Synchrones Umschließen
(Wrapping)

Synchrones Umschließen

- Synchron
- Blocks werden nur auf Stack angelegt
- Mitgegebener Code wird mit Logik umschlossen
- Sehr effizient
- Ersatz für Compiler-Makros

Synchrones Umschließen

```
typedef void (^MCNActionBlock)();

@implementation MCNTools

+ (void)timeAction:(NSString*)actionName block:(MCNActionBlock)block {
    NSParameterAssert(actionName);
    NSParameterAssert(block);
    NSDate* startDate = [NSDate date];
    block();
    NSTimeInterval duration = -startDate.timeIntervalSinceNow;
    NSLog(@"%@ took %gs", actionName, duration);
}

@end
```

Enumerieren

Enumerieren

```
NSArray* names = [NSArray arrayWithObjects:@“Leonard“, @“Sheldon“,
                                           @“Howard“,  @“Rajesh“, nil];

[names enumerateObjectsWithOptions: NSEnumerationReverse
                        usingBlock:^(id obj, NSUInteger idx, BOOL *stop) {
  NSString* iString = obj;
  NSLog(@“%@“, iString);
  if([iString isEqualToString:@“Sheldon“])
    *stop = YES;
}];

/* Output: Rajesh Howard Sheldon */
```

Enumerieren

```
NSDictionary* dict = [NSDictionary
dictionaryWithObjectsAndKeys:@“Physicist“, @“Sheldon“,
                             @“Engineer“,  @“Howard“, nil];

[dict enumerateKeysAndObjectsUsingBlock:^(id key, id obj, BOOL *stop) {
  NSLog(@“%@ -> %@“, key, obj);
}];

/* Output: Sheldon -> Physicist
           Howard  -> Engineer */
```

Enumerieren

```
NSMutableIndexSet* set = [NSMutableIndexSet indexSet];
[set addIndex:42];
[set addIndexesInRange:NSMakeRange(5,3)];

[set enumerateIndexesInRange:NSMakeRange(6,50)
                     options:NSEnumerationReverse
                  usingBlock:^(NSUInteger index, BOOL *stop) {
  NSLog(@“%d“,index);
}];

/* Output: 42 7 6*/
```

Enumerieren

- `-[NSSet enumerateObjectsWithOptions:usingBlock]`
- `-[NSAttributedString enumerateAttributesInRange:options:usingBlock:]`
- `NSEnumerationConcurrent`
- Keine Unterstützung von NSMapTable & NSHashTable (Mac)
  - Können in Kategorien implementiert werden

Enumerieren

```
@implementation NSMapTable (MacounAdditions)

- (void)enumerateKeysAndObjectsUsingBlock:(void (^)(id key, id obj,
                                            BOOL *stop)) block {
    NSParameterAssert(block != nil);
    for(NSString* key in self) {
        BOOL shouldStop = NO;
        block(key, [self objectForKey:key], &shouldStop);
        if(shouldStop)
            break;
    }
}

@end
```

Filtern

Filtern

```
@interface Nerd : NSObject
@property (copy) NSString* name;
- (id)initWithName:(NSString*)name;
@end

@implementation Nerd
@synthesize name = name_;

- (id)initWithName:(NSString*)name {
  if((self = [self init])) {
    name_ = [name copy];
  }
  return self;
}
@end
```

Filtern

```
Nerd* leonard = [[Nerd alloc] initWithName:@“Leonard“];
Nerd* sheldon = [[Nerd alloc] initWithName:@“Sheldon“];
Nerd* rajesh  = [[Nerd alloc] initWithName:@“Rajesh“];
Nerd* howard  = [[Nerd alloc] initWithName:@“Howard“];
NSArray* nerds = [NSArray arrayWithObjects: leonard, sheldon,
                                            rajesh, howard, nil];

NSIndexSet* matches = [nerds indexesOfObjectsPassingTest:^BOOL(id obj,
                                          NSUInteger idx, BOOL *stop) {
  Nerd* iNerd = obj;
  return iNerd.name.length == 7;
}];

/* matches: 0-1 */
```

Filtern

```
NSPredicate* predicate = [NSPredicate predicateWithBlock:^BOOL(id obj,
                                                              NSDictionary* bindings) {
  Nerd* iNerd = obj;
  return iNerd.name.length == 6;
}];

NSArray* matches = [nerds filteredArrayUsingPredicate:predicate];

/* matches: rajesh, howard */
```

Sortieren

```
NSArray* sortedNerds = [nerds sortedArrayWithOptions:NSSortStable
                                          usingComparator:
  ^NSComparisonResult(id o1, id o2) {
  NSUInteger len1 = ((Nerd*)o1).name.length;
  NSUInteger len2 = ((Nerd*)o2).name.length;
  return len1 < len2 ? NSOrderedAscending :
         len1 > len2 ? NSOrderedDescending : NSOrderedSame;
}];

/* sortedNerds: rajesh, howard, leonard, sheldon */
```

Demo

Funktionen
Höherer Ordnung

Funktionen Höherer Ordnung

```
NSArray* names = [nerds map:^BOOL(id obj) {
     return ((Nerd*)obj).name;
}];

// @“Leonard“, @“Sheldon“, @“Rajesh“, @“Howard“
```

```
BOOL allNamesShort = [nerds all:^BOOL(id obj) {
     return ((Nerd*)obj).name.length < 10;
}];

// YES
```

Funktionen Höherer Ordnung

```
- (BOOL)all:(BOOL (^)(id))block {
    NSParameterAssert(block);
    for (id obj in self)
       if(!block(obj))
           return NO;
    return YES;
}
```

```
- (BOOL)any:(BOOL (^)(id))block {
    NSParameterAssert(block);
    for (id obj in self)
        if(block(obj))
            return YES;
    return NO;
}
```

Funktionen Höherer Ordnung

```
- (NSArray*)map:(id (^)(id))block {
    NSParameterAssert(block);
    NSMutableArray* new = [NSMutableArray arrayWithCapacity:self.count];
    for (id obj in self) {
        id mapped = block(obj);
        [new addObject: mapped ? mapped : [NSNull null]];
    }
    return [new copy];
}
```

Funktionen Höherer Ordnung

```
@interface NSArray (MacounAdditions)
- (BOOL)all:(BOOL (^)(id obj))block;
- (BOOL)any:(BOOL (^)(id obj))block;
- (id)match:(BOOL (^)(id obj))block;
- (NSArray*)select:(BOOL (^)(id obj))block;
- (void)partitionIntoMatches:(NSArray**)outMatches
                     misses:(NSArray**)outMisses
                      block:(BOOL (^)(id obj))block;
- (NSArray*)map:(id (^)(id obj))block;
- (NSArray*)mapWithoutNull:(id (^)(id))block;
@end
```

Asynchroner Aufruf

Asynchroner Aufruf

- Alternative zu Delegation-Entwurfsmuster
- Vorteile
  - lineares Aufschreiben
  - keine sperrigen void*- oder NSDictionary-Kontexte
  - keine Ivars

Lineares Aufschreiben

```
@implementation MyViewController

- (IBAction)export:(id)sender
{
   NSURL* URL = ... ; // might be nil

   [self exportToURL:URL
   completionHandler:^(BOOL success, NSError* error) {
       if(error)
           [self presentError:error];
   }];
}

@end
```

Lineares Aufschreiben

```
@interface NSSavePanel

- (void)beginSheetModalForWindow:(NSWindow *)window
                completionHandler:(void (^)(NSInteger result))handler;

@end
```

```
- (void)exportToURL:(NSURL*)URL
  completionHandler:(void (^)(BOOL success, NSError* error))handler
{
  NSParameterAssert(handler);

  __block NSError* error = nil;
  if(!URL) {
    NSSavePanel* panel = [NSSavePanel savePanel];
    [panel beginSheetModalForWindow:self.view.window
                completionHandler:^(NSInteger result) {
                  if(result == NSFileHandlingPanelOKButton)
                    handler([self exportToURL:panel.URL error:&error], error);
                }];
  }
  else
    handler([self exportToURL:URL error:&error], error);
}
```

Alerts

Alerts

```
- (void)deleteNerd:(Nerd*)nerd
{
    NSAlert* alert = [[NSAlert alloc] init];
    alert.messageText = [NSString stringWithFormat:
                         @"Do you really want to delete %@?", nerd.name];
    [alert addButtonWithTitle:@"Delete"];
    [alert addButtonWithTitle:@"Cancel"];
    [alert beginSheetModalForWindow:self.view.window
                completionHandler:^(NSInteger result) {
                    if(result == NSAlertFirstButtonReturn)
                        [self doDeleteNerd:nerd];
                }];
}
```

Alerts

```
@interface NSAlert (MacounAdditions)

- (void)beginSheetModalForWindow:(NSWindow*)window
                completionHandler:(void (^)(NSInteger result))handler;

@end
```

Alerts

```
typedef void (^ModalBlock)(NSInteger resultCode);

@implementation NSAlert (MacounAdditions)

- (void)beginSheetModalForWindow:(NSWindow*)window
                completionHandler:(void (^)(NSInteger result))handler
{
    [self beginSheetModalForWindow:window
                      modalDelegate:self
                     didEndSelector:@selector(alertDidEnd:code:context:)
                        contextInfo:CFRetain([handler copy])];
}

// ...
```

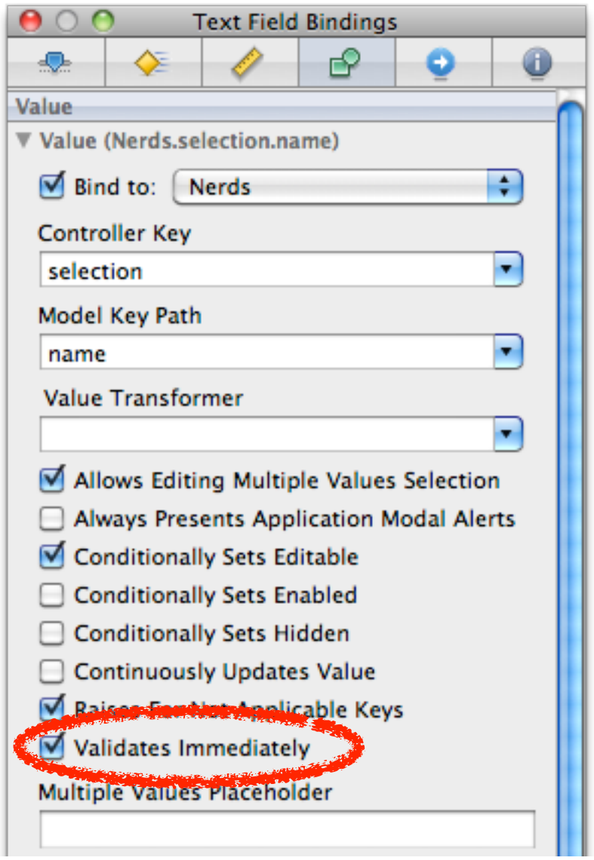
Alerts

```
// ...

- (void)alertDidEnd:(NSAlert*)alert
                code:(NSInteger)code
             context:(void*)context
{
    [alert.window close];
    ModalBlock handler = (ModalBlock)context;
    handler(code);
    CFRelease(handler);
}

@end
```

Key-Value-Validation

Key-Value-Validation

```
@protocol NSKeyValueCoding
// ...

- (BOOL)validateValue:(id*)ioValue
                forKey:(NSString*)key
                 error:(NSError**)outError;

// ...
@end
```



- Sucht nach

  `-(BOOL)validate<Key>:(id*)ioVal error:(NSError**)outError;`

Key-Value-Validation

```
@interface Nerd
// ...

- (BOOL)validateName:(NSString**)inOutName
                error:(NSError **)outError;

// ...
@end
```

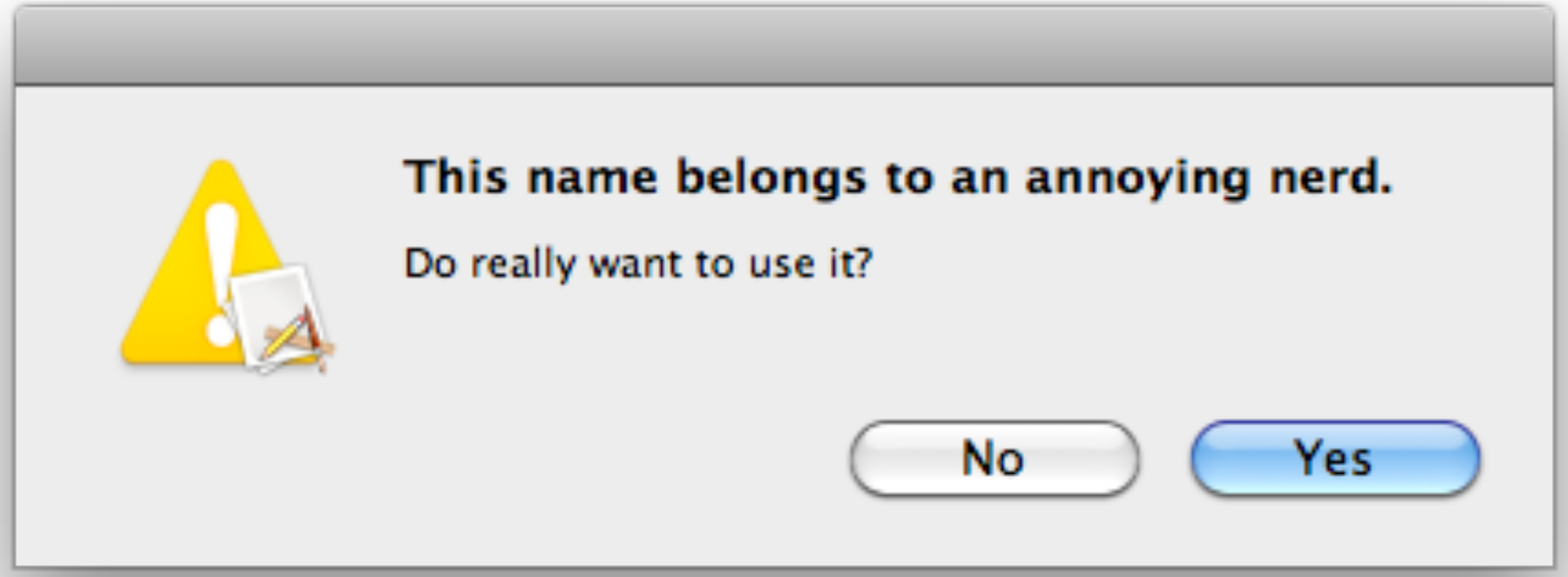
```
- (BOOL)validateName:(NSString**)name error:(NSError**)err
{
  if([*name isEqualToString:@“Sheldon“]) {
    if(err)
      *err = [NSError errorWithDomain:@“de.Macoun“
                                 code:1234
                              message:@“This name belongs to an annoying nerd.“
                     recoverySuggestion:@“Do really want to use it?“
                              options:[NSArray arrayWithObjects:@“Yes“, @“No“, nil]
                                block:^(NSError* e, NSUInteger optionIndex) {
                                   if(optionIndex == 0) {
                                      self.name = *name;
                                      return YES;
                                   }
                                   return NO;
                                }];
    return NO;
  }
  return YES;
}
```

```
typedef BOOL (^MCNErrorRecoveryBlock)(NSError* error, NSUInteger optionIndex);
static NSString* const RecoveryBlockKey = @"de.macoun.NSError.recoveryBlock";

@implementation NSError (MacounAdditions)
+ (NSError*)errorWithDomain:(NSString*)domain
                       code:(NSInteger)code
                    message:(NSString*)message
          recoverySuggestion:(NSString*)recoverySuggestion
                    options:(NSArray*)options
                      block:(MCNErrorRecoveryBlock)block
{
  NSMutableDictionary* info = [NSMutableDictionary dictionary];
  [info setValue:message            forKey:NSLocalizedDescriptionKey];
  [info setValue:recoverySuggestion forKey:NSLocalizedRecoverySuggestionErrorKey];
  [info setObject:options           forKey:NSLocalizedRecoveryOptionsErrorKey];
  [info setObject:self              forKey:NSRecoveryAttempterErrorKey];
  [info setObject:[block copy]      forKey:RecoveryBlockKey];
  return [NSError errorWithDomain:domain code:code userInfo:info];
}
// ...
```

```
// ...

+ (BOOL)attemptRecoveryFromError:(NSError*)error
                    optionIndex:(NSUInteger)optionIndex
{
    MCNErrorRecoveryBlock block = [error.userInfo objectForKey:RecoveryBlockKey];
    return block(error, optionIndex);
}

// ...
```

```
// ...
+ (void)attemptRecoveryFromError:(NSError*)error
                     optionIndex:(NSUInteger)optionIndex
                        delegate:(id)delegate
              didRecoverSelector:(SEL)selector
                     contextInfo:(void*)contextInfo
{
    MCNErrorRecoveryBlock block = [error.userInfo objectForKey:RecoveryBlockKey];
    BOOL recovered = block(error, optionIndex);
    NSMethodSignature* sig = [delegate methodSignatureForSelector:selector];
    NSInvocation* invocation = [NSInvocation invocationWithMethodSignature:sig];
    invocation.selector = selector;
    invocation.target = delegate;
    [invocation setArgument:&recovered atIndex:2];
    [invocation setArgument:&contextInfo atIndex:3];
    [invocation invoke];
}
@end
```

- `-[NSResponder presentError:]`
- `-[NSResponder presentError:modalForWindow:delegate:didPresentSelector:contextInfo:]`



Alternative zum Ableiten von Klassen

Alternative zum Ableiten

Formatter

```
@interface MCNBlockFormatter : NSFormatter

@property (copy)  NSString* (^toStringBlock)(id object);
@property (copy)  BOOL      (^toObjectBlock)(id* object, NSString* string,
                                             NSString** errorDesc);

@end
```

Alternative zum Ableiten

Formatter

```
@implementation MCNBlockFormatter
@synthesize toStringBlock;
@synthesize toObjectBlock;

- (NSString*)stringForObjectValue:(id)anObject {
    return toStringBlock(anObject);
}

- (BOOL)getObjectValue:(id*)anObject
              forString:(NSString*)string
      errorDescription:(NSString**)error {
        return toObjectBlock(anObject, string, error);
}
@end
```

Alternative zum Ableiten

Formatter

```
PWBlockFormatter* cheapURLFormatter = [[MCNBlockFormatter alloc] init];

formatter.toStringBlock = ^(id value) {
  return ((NSURL*)value).absoluteString;
};

formatter.toObjectBlock = ^(id* value, NSString* string, NSString** err) {
  if(string == nil) {
    *value = nil;
    return YES;
  }
  *value = [NSURL URLWithString:string];
  return (*value) != nil;
};
```

Alternative zum Ableiten

Einfache Views

```
MCNBlockView* backgroundView = [[MCNBlockView alloc]
       initWithFrame:NSInsetRect(self.view.bounds, 5.0, 5.0)
           drawBlock:^(MCNBlockView* bView, NSRect dirtyRect) {
                        [[NSColor grayColor] set];
                        NSRectFill(dirtyRect);
                        [[NSColor redColor] set];
                        [NSBezierPath strokeRect:view.bounds];
                      }];

[self.view addSubview:backgroundView];
```

Alternative zum Ableiten

Einfache Views

```
@class MCNBlockView;

typedef void(^MCNDrawBlock)(MCNBlockView* view, NSRect dirtyRect);

@interface MCNBlockView : NSView

- (MCNBlockView*)initWithFrame:(NSRect)frame
                     drawBlock:(MCNDrawBlock)block;

@property (readonly) MCNDrawBlock drawBlock;

@end
```

Alternative zum Ableiten

```
@implementation MCNBlockView
@synthesize drawBlock;

- (MCNBlockView*)initWithFrame:(NSRect)frame
                     drawBlock:(MCNDrawBlock)block {
   NSParameterAssert(block);
   if (self = [self initWithFrame:frame])
      drawBlock = [block copy];
   return self;
}

- (void)drawRect:(NSRect)dirtyRect {
   drawBlock(self, dirtyRect);
}
@end
```

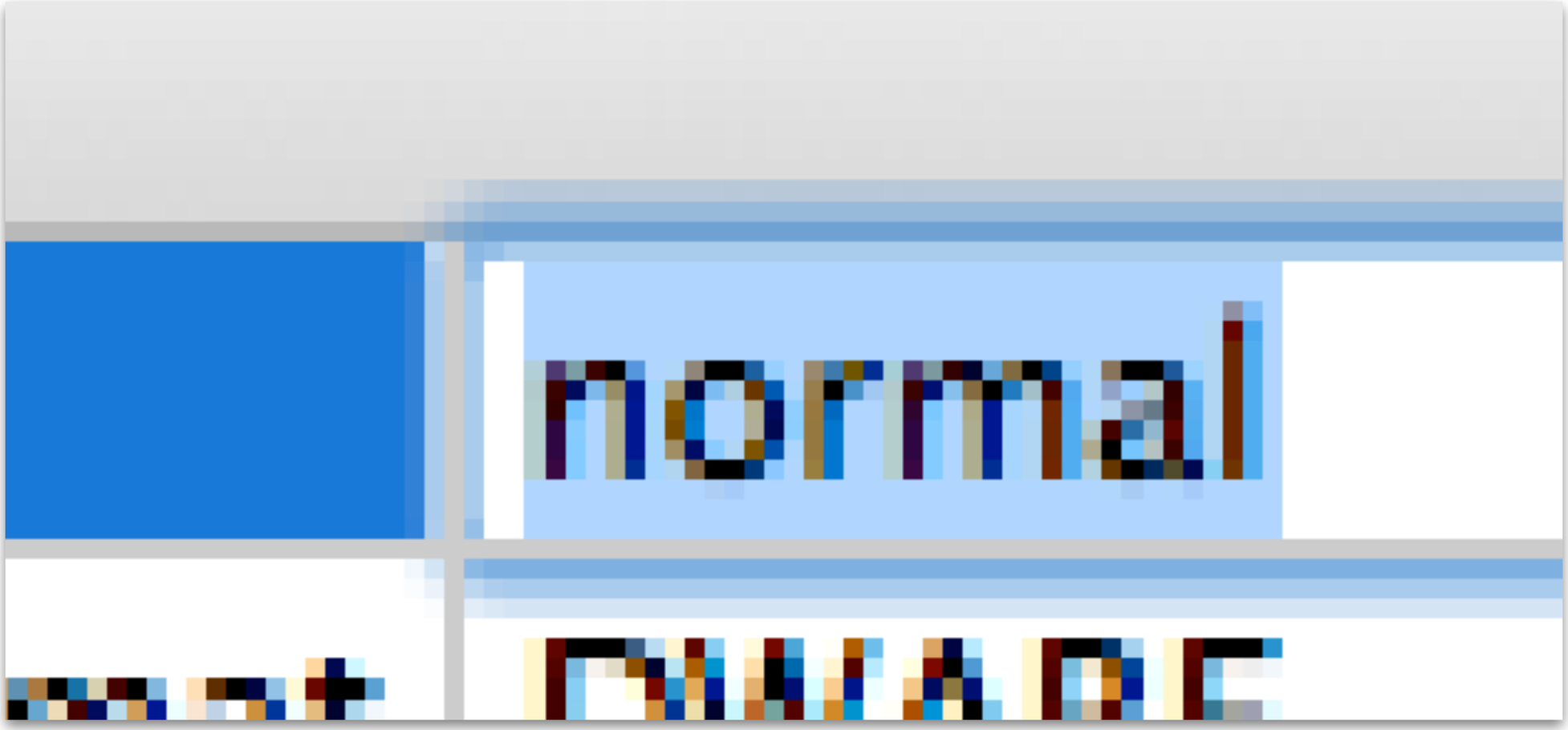
Fokusringe

Xcode

Setting	Value
Per-configuration Inter...	/Users/frank/Projekte/Builds...
Precompiled Headers C...	/var/folders/5x/5x8Qo87xH...
▼Build Options	
Build Variants	normal
Debug Information Format	DWARF
Enable OpenMP Support	☐
Generate Profiling Code	☐
Precompiled Header Us...	☑
Run Static Analyzer	☐
Scan All Source Files for...	☐
Validate Built Product	☐

Fokusringe

Xcode



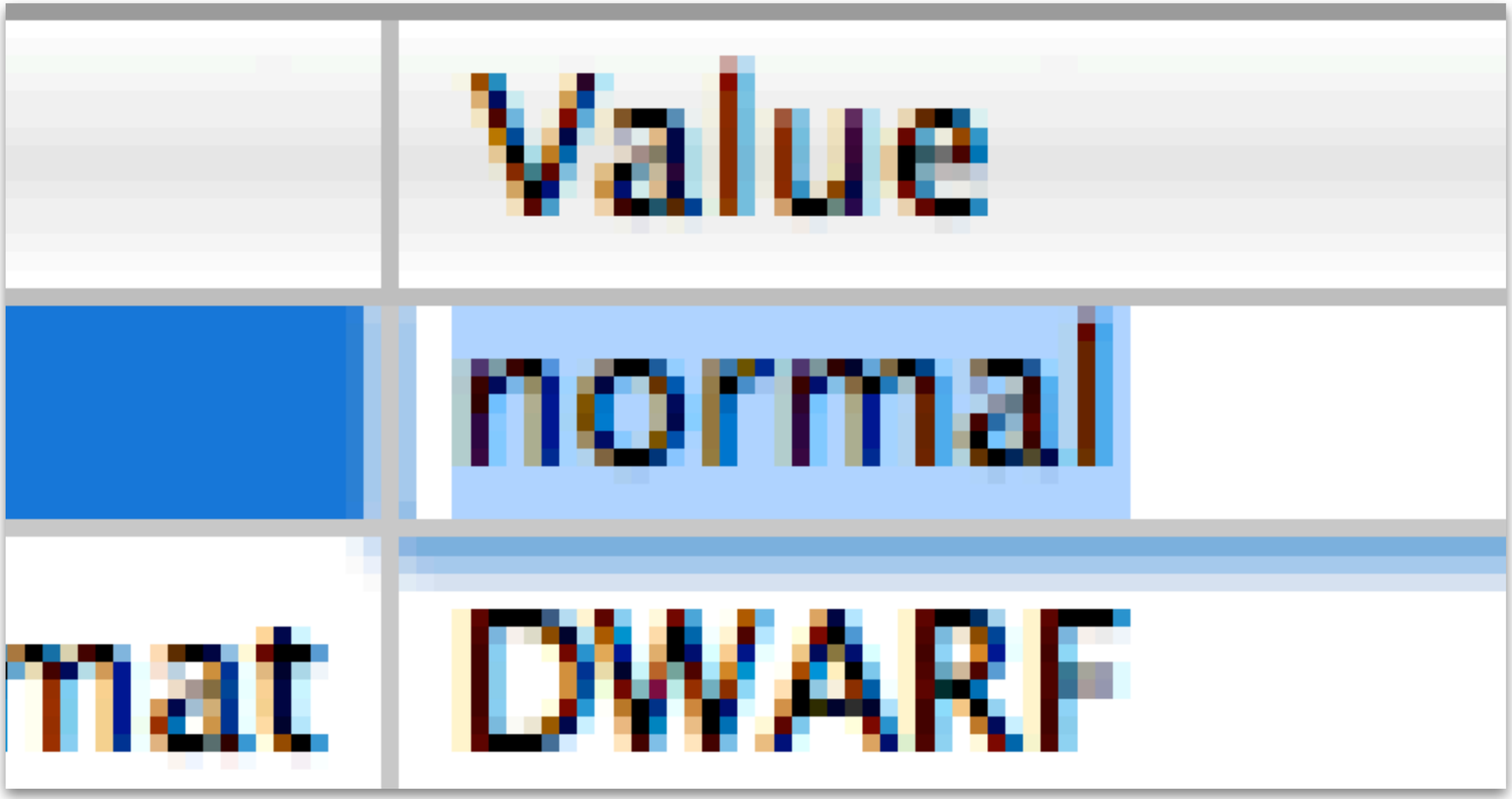
Fokusringe

Xcode

Setting	Value
Build Variants	normal
Debug Information Format	DWARF
Enable OpenMP Support	☐
Generate Profiling Code	☐
Precompiled Header Us...	☑
Run Static Analyzer	☐
Scan All Source Files for...	☐
Validate Built Product	☐
▼Code Signing	
Code Signing Entitlements	
Code Signing Identity	
Code Signing Resource ...	
Other Code Signing Flags	

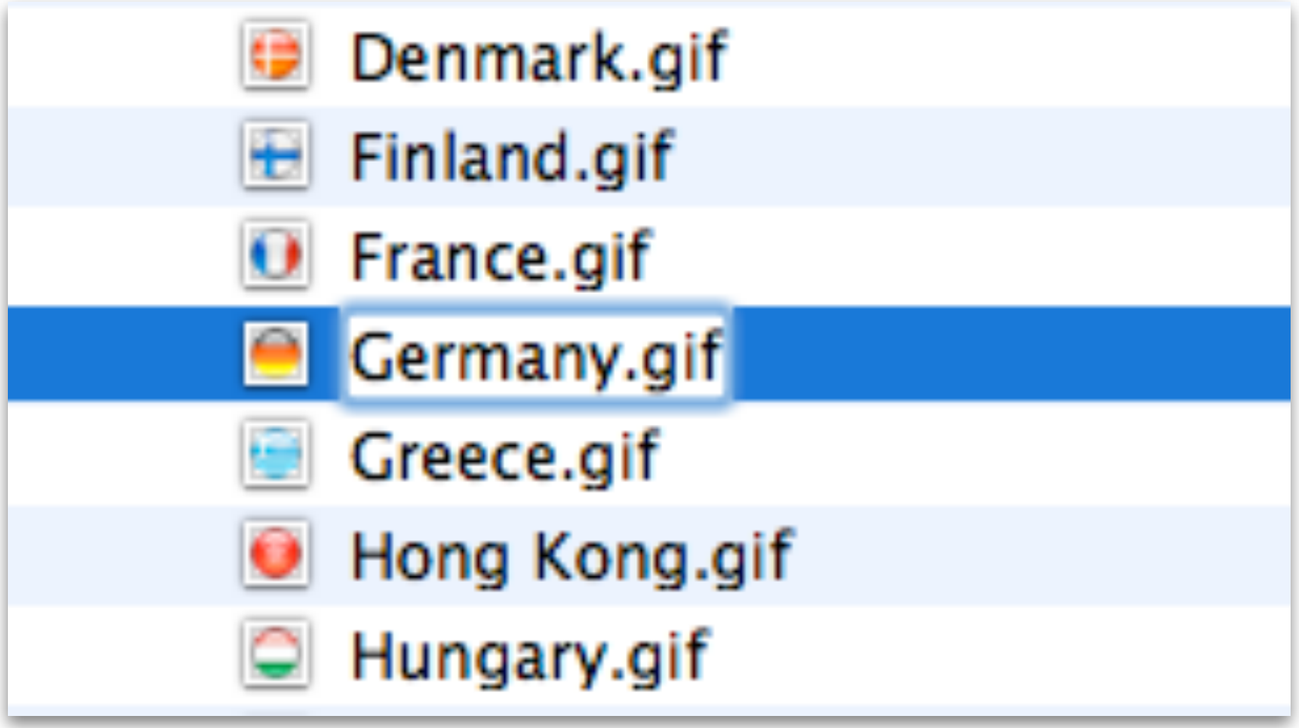
Fokusringe

Xcode



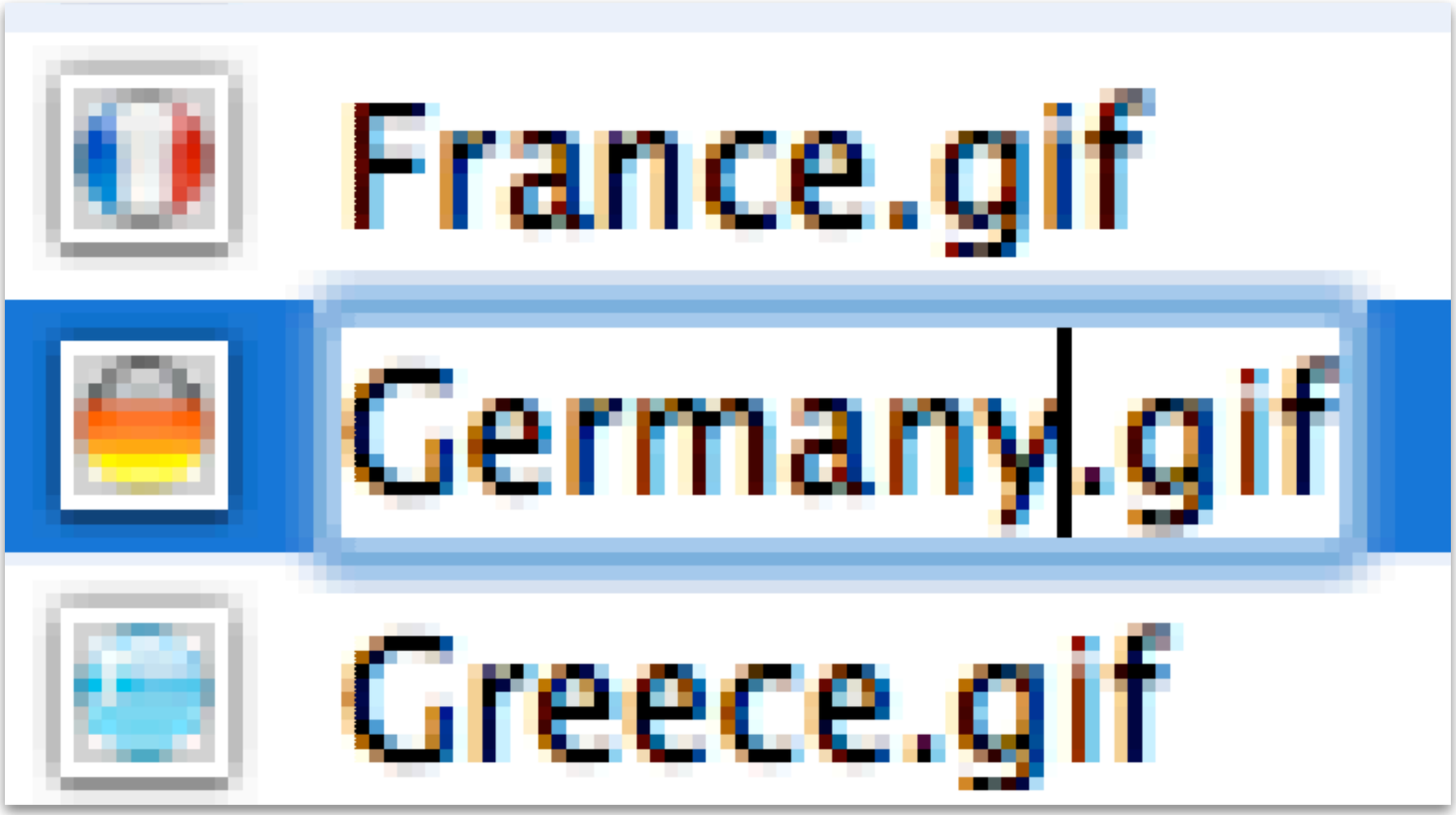
Fokusringe

Finder



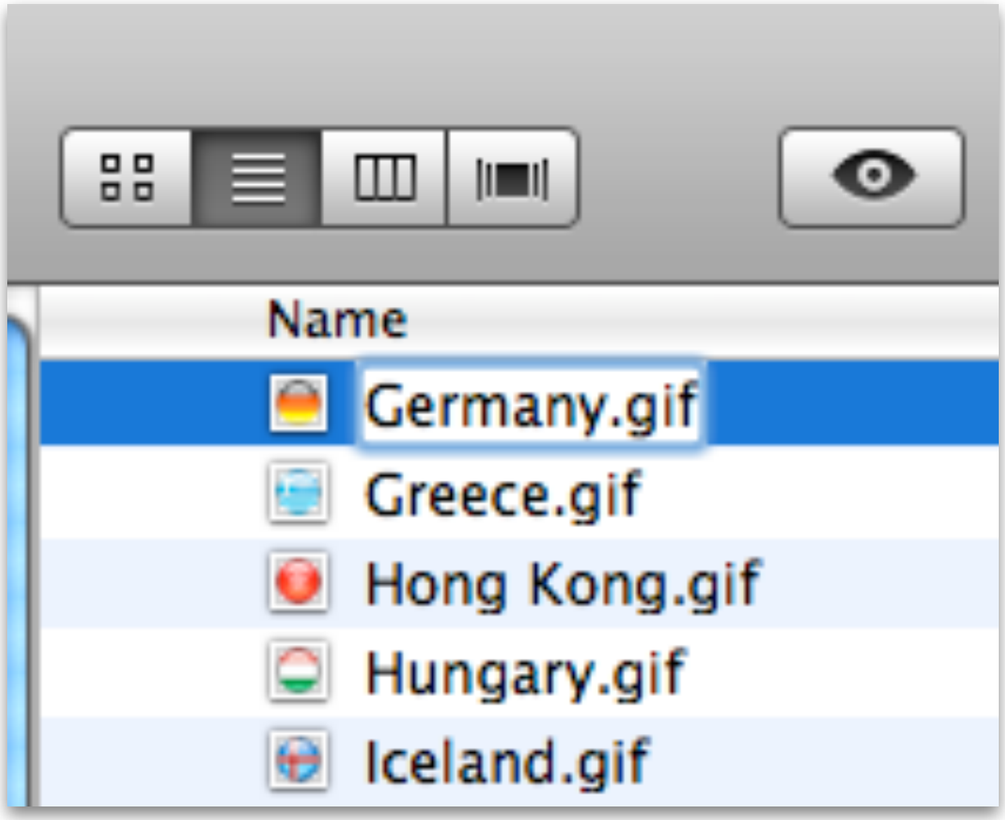
Fokusringe

Finder



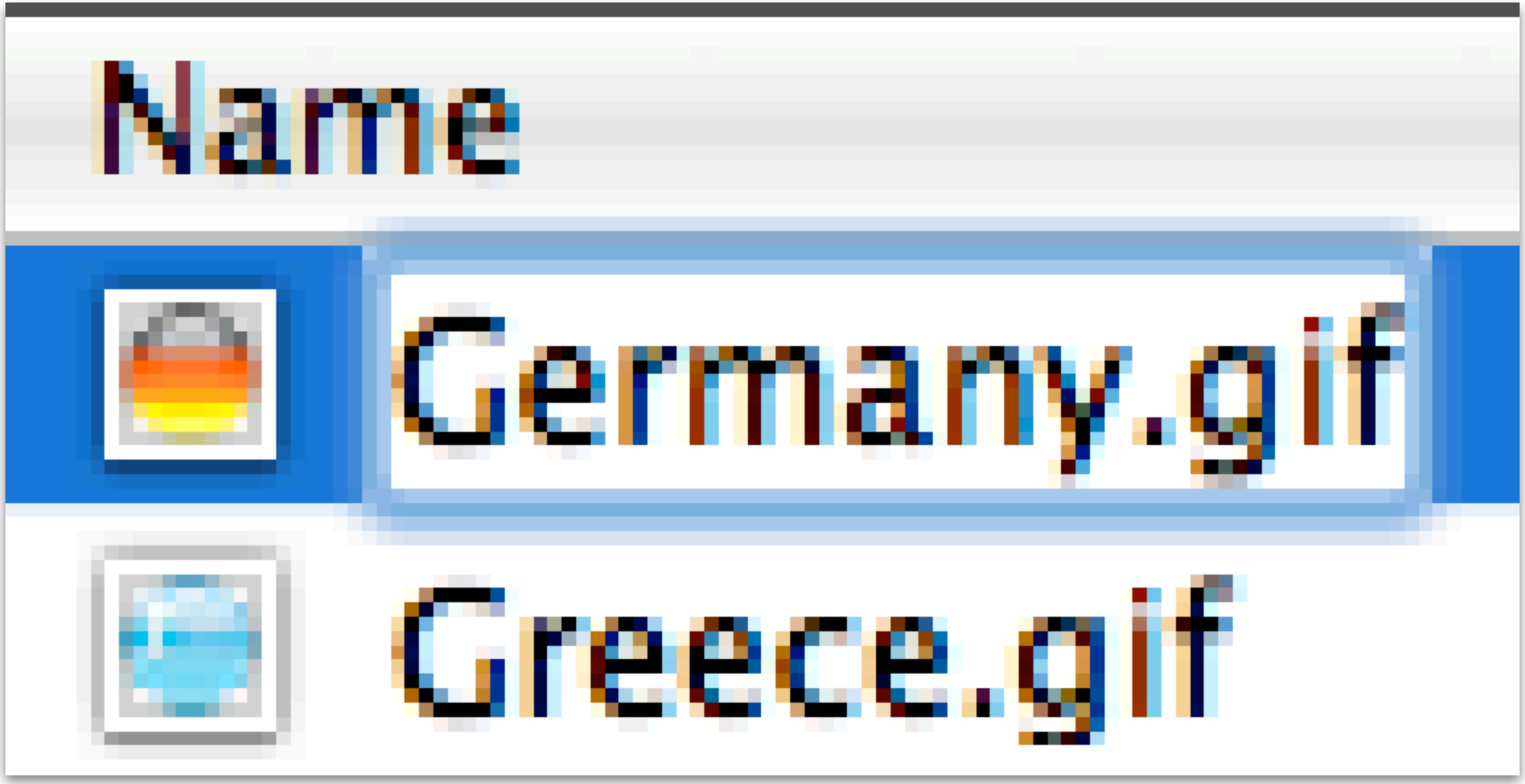
Fokusringe

Finder am oberen Rand



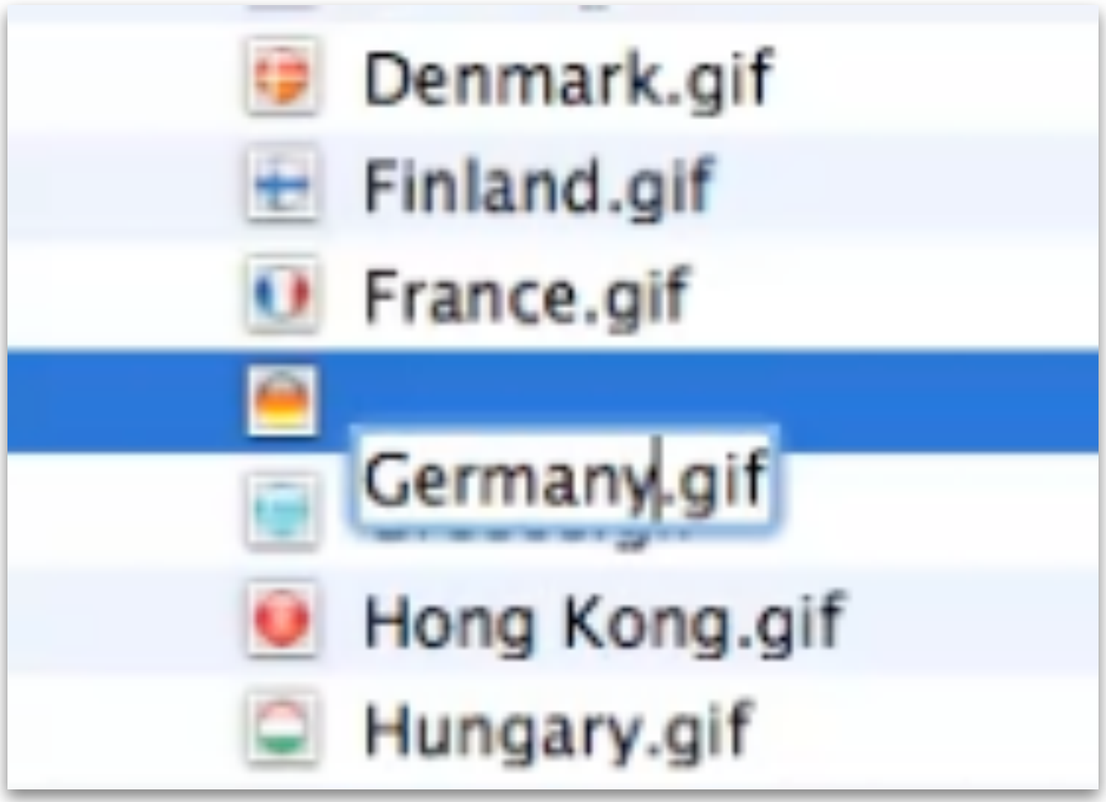
Fokusringe

Finder am oberen Rand



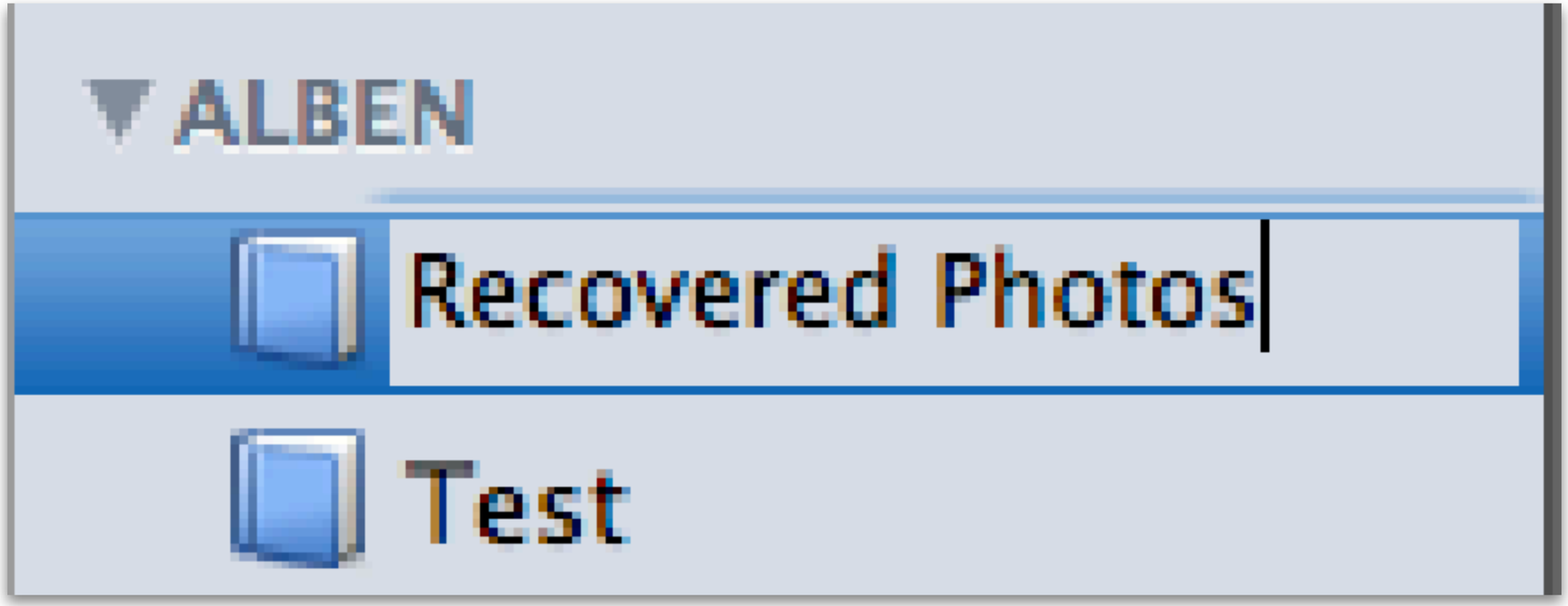
Fokusringe

Finder beim Bewegen



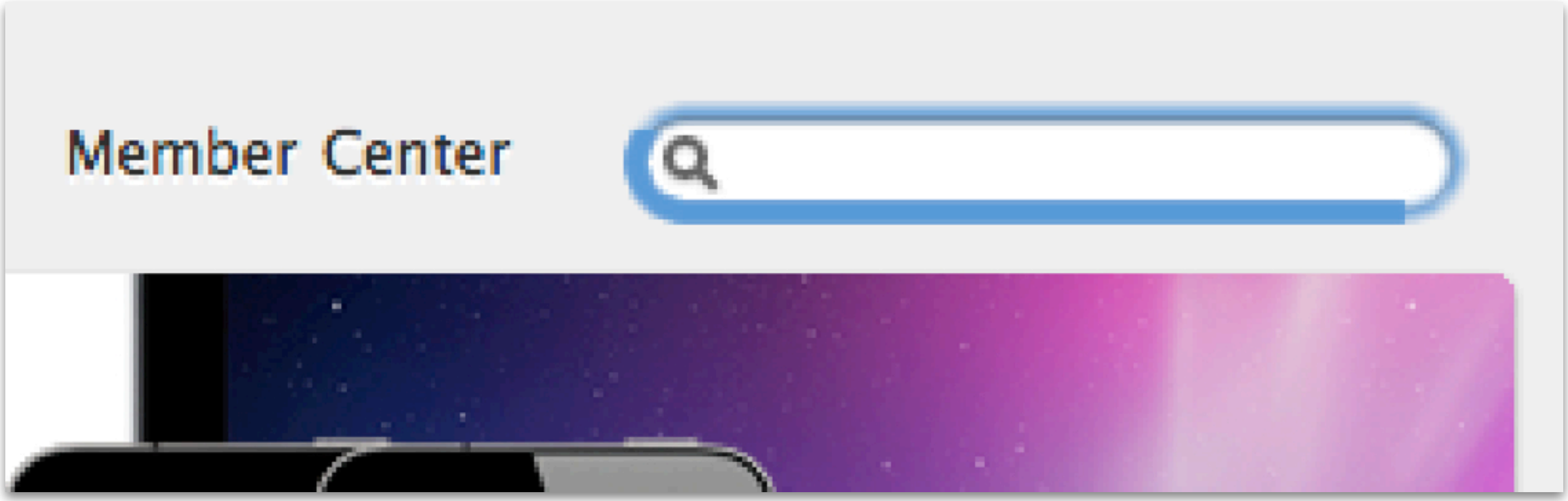
Fokusringe

iPhoto



Fokusringe

Safari

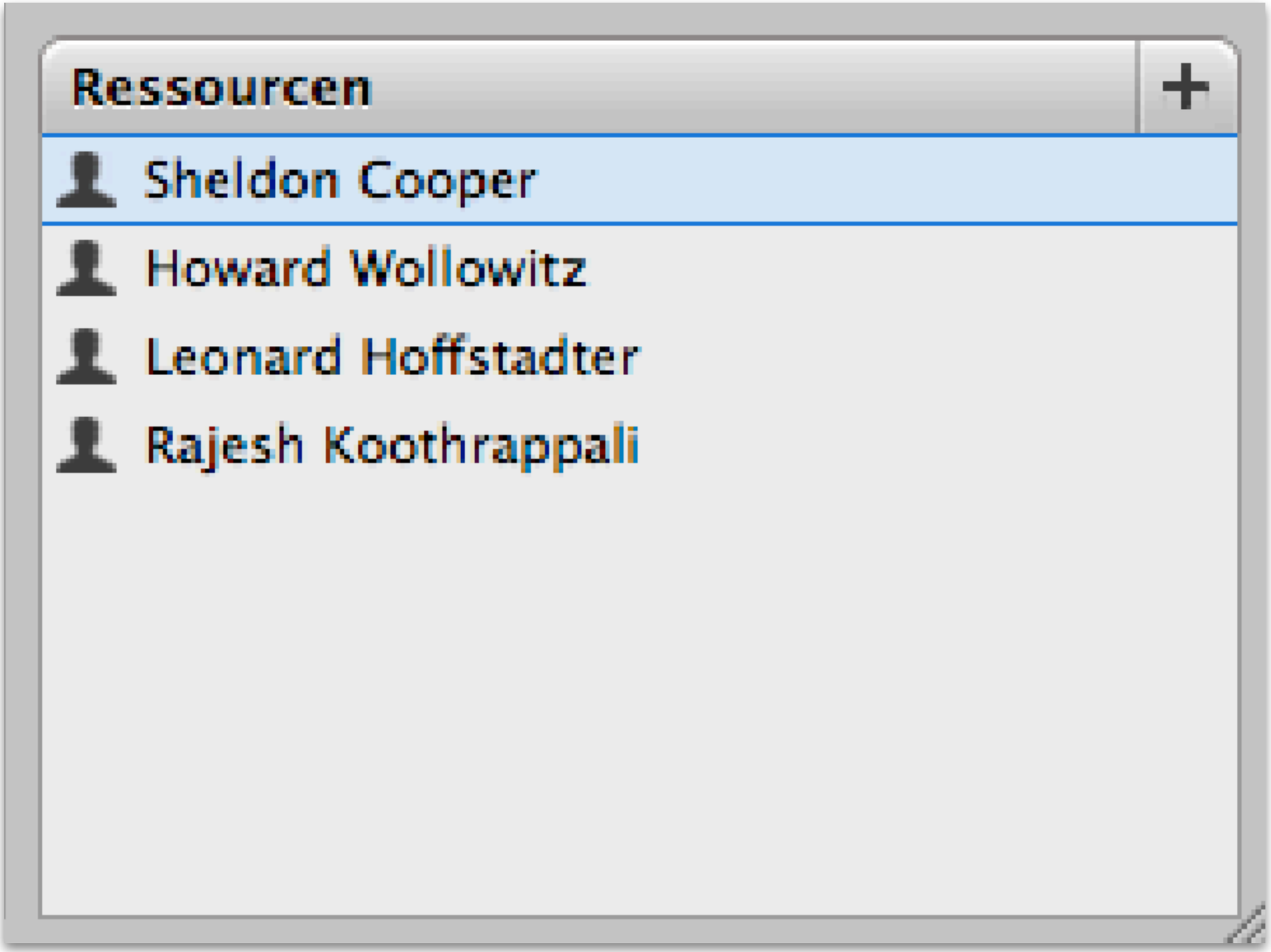


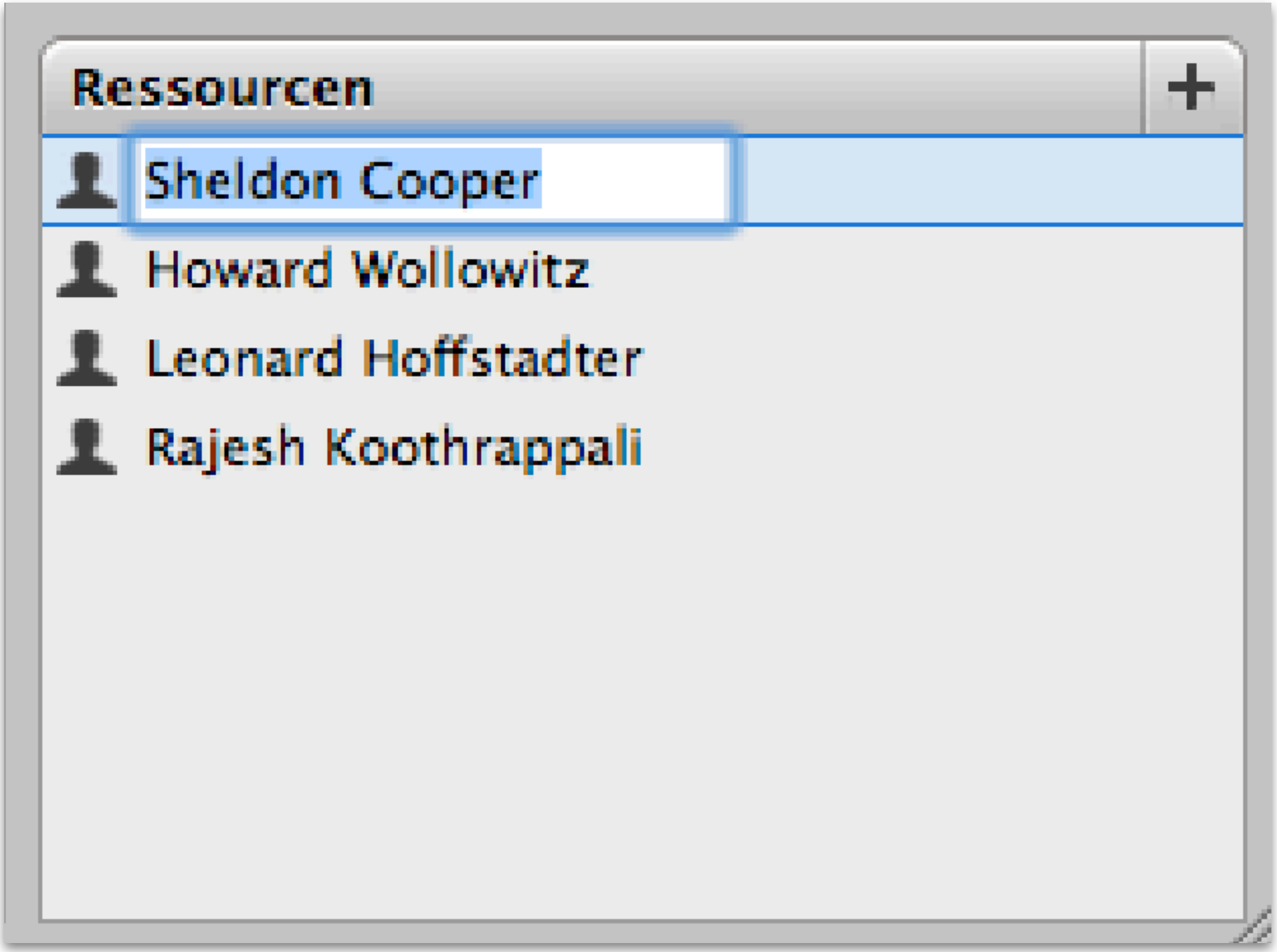
Unsere Lösung

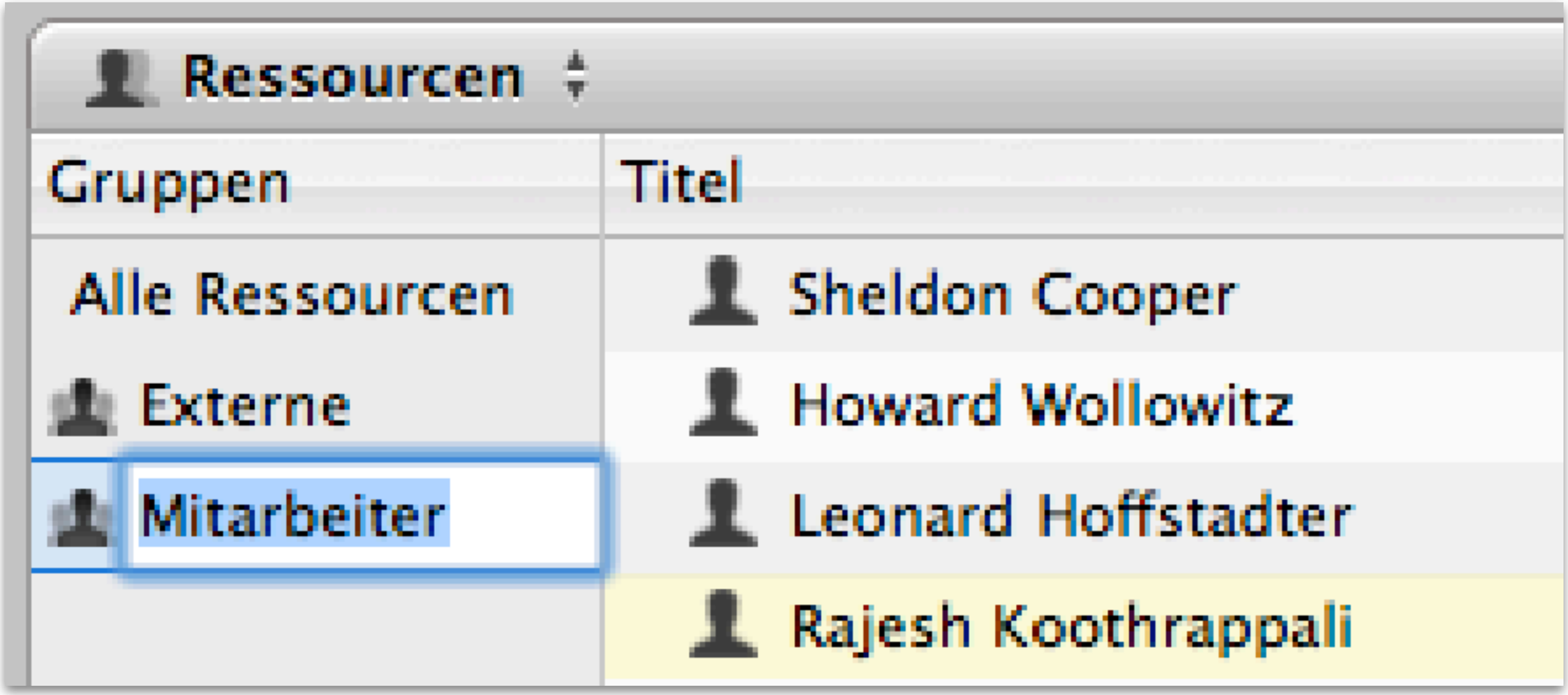
- Transparente Overlay-View über das gesamte Fenster
- Overlay-View ist letzte Subview der Window-Content-View
- Views die Overlays brauchen registrieren diese beim Fenster
- Zeichen- und Positionierungslogik werden als Block angehängt

```
typedef void(^PWOverlayRectUpdater)(PWOverlay* overlay);
typedef void(^PWOverlayDrawer)(PWOverlay* overlay, CGRect dirtyRect);

@interface PWOverlay : NSObject
@property (readonly)    NSView*                referenceView;
@property (readonly)    NSView*                overlayView;
@property (copy)        PWOverlayRectUpdater rectUpdater;
@property (copy)        PWOverlayDrawer        drawer;
@property (readwrite)   CGRect                 rect;
@property (readwrite)   PWInteger              layer;
@end
```

```
- (void) createSelectionTopStrokeOverlayForRow:(PWOutlineRow*)row
{
  if (!rowSelectionLineOverlay_) {
    rowSelectionLineOverlay_ = [self.window createOverlayForView:self.bodyView];
    rowSelectionLineOverlay_.drawer = ^(PWOverlay* overlay, CGRect dirtyRect) {
      PWGraphicsContext* context = [overlay.overlayView contextForDrawRect];
      CGContextRef ctx = context.cgContext;
      CGContextSaveGState (ctx);

      CGRect lineRect = CGRectInset (overlay.rect, 1.0, 1.0);
      CGPoint p1 = CGPointMake (CGRectGetMinX (lineRect), CGRectGetMinY (lineRect));
      CGPoint p2 = CGPointMake (CGRectGetMaxX (lineRect), p1.y);
      CGSize offsets = { 0.0, 0.5 };
      p1 = PWSnapPointToPixel (ctx, p1, PWSnapToFullPixel, PWSnapToFullPixel, &offsets);
      p2 = PWSnapPointToPixel (ctx, p2, PWSnapToFullPixel, PWSnapToFullPixel, &offsets);
      CGContextSetStrokeColorWithColor (ctx, self.viewController.selectedRowStrokeColor);
      CGContextBeginPath      (ctx);
      CGContextSetLineWidth   (ctx, 1.0);
      CGContextMoveToPoint    (ctx, p1.x, p1.y);
      CGContextAddLineToPoint (ctx, p2.x, p2.y);
      CGContextStrokePath     (ctx);
      CGContextRestoreGState  (ctx);
};
```


Ressourcen
Sheldon Cooper
Howard Wollowitz
Leonard Hoffstadter
Rajesh Koothrappali


Ressourcen
Sheldon Cooper
Howard Wollowitz
Leonard Hoffstadter
Rajesh Koothrappali


Ressourcen
Gruppen
Titel
Alle Ressourcen
Externe
Mitarbeiter
Sheldon Cooper
Howard Wollowitz
Leonard Hoffstadter
Rajesh Koothrappali

Fragen?

Vielen Dank

Macoun'10