跟谁学 iOS 技术培训 之

Objective-C 基础

2016-07 明林清

Objective-C != Object Oriented C

- 语言
- 语法
- ~~Frameworks~~
- ~~Runtime~~
- ...

- Class
- object
- method
- ivar (instance variable)
- property
- block

Class

Class

```objc
//  BJWebSocketClient.h

#import <Foundation/Foundation.h>

@interface BJWebSocketClient : NSObject

@property (nonatomic, readonly) BJWSCState state;
@property (nonatomic, weak) id<BJWebSocketClientDelegate> delegate;

- (void)connect;
- (void)disconnectWithReason:(BJWSCDisconnectReason)reason;

- (void)sendMessage:(NSString *)message;

@end
```

Class implementation

```
//  BJWebSocketClient.m

#import "BJWebSocketClient.h"
#import "BJWebSocketClient+BJProtected.h"

@implementation BJWebSocketClient

// ...

@end
```

Class Extension

```
//  BJWebSocketClient+BJProtected.h
#import "BJWebSocketClient.h"

@interface BJWebSocketClient ()

@property (nonatomic, readwrite) BJWSCState state;

@end
```

Class Extension

- private
- protected

Category

```
//  BJWebSocketClient+BJEmoticonMessage.h

#import "BJWebSocketClient.h"

@interface BJWebSocketClient (BJEmoticonMessage)

// ???: @property

- (void)sendEmoticon:(NSString *)emoticon;

@end
```

Category implementation

```m
//  BJWebSocketClient+BJEmoticonMessage.m

#import "BJWebSocketClient+BJEmoticonMessage.h"

@implementation BJWebSocketClient (BJEmoticonMessage)

- (void)sendEmoticon:(NSString *)emoticon {
    NSString *message = [self messageWithEmoticon:emoticon];
    [self sendMessage:message];
}

- (NSString *)messageWithEmoticon:(NSString *)emoticon {
    return [NSString stringWithFormat:@"<<%@>>", emoticon];
}

@end
```

Protocol

```objc
@protocol BJWebSocketClientDelegate <NSObject>

// ???: @property

@required

- (void)webSocketClient:(BJWebSocketClient *)webSocketClient
         stateDidChange:(BJWSCState)state;

@optional

- (void)webSocketClient:(BJWebSocketClient *)webSocketClient
         didSendMessage:(NSString *)message;

@end
```

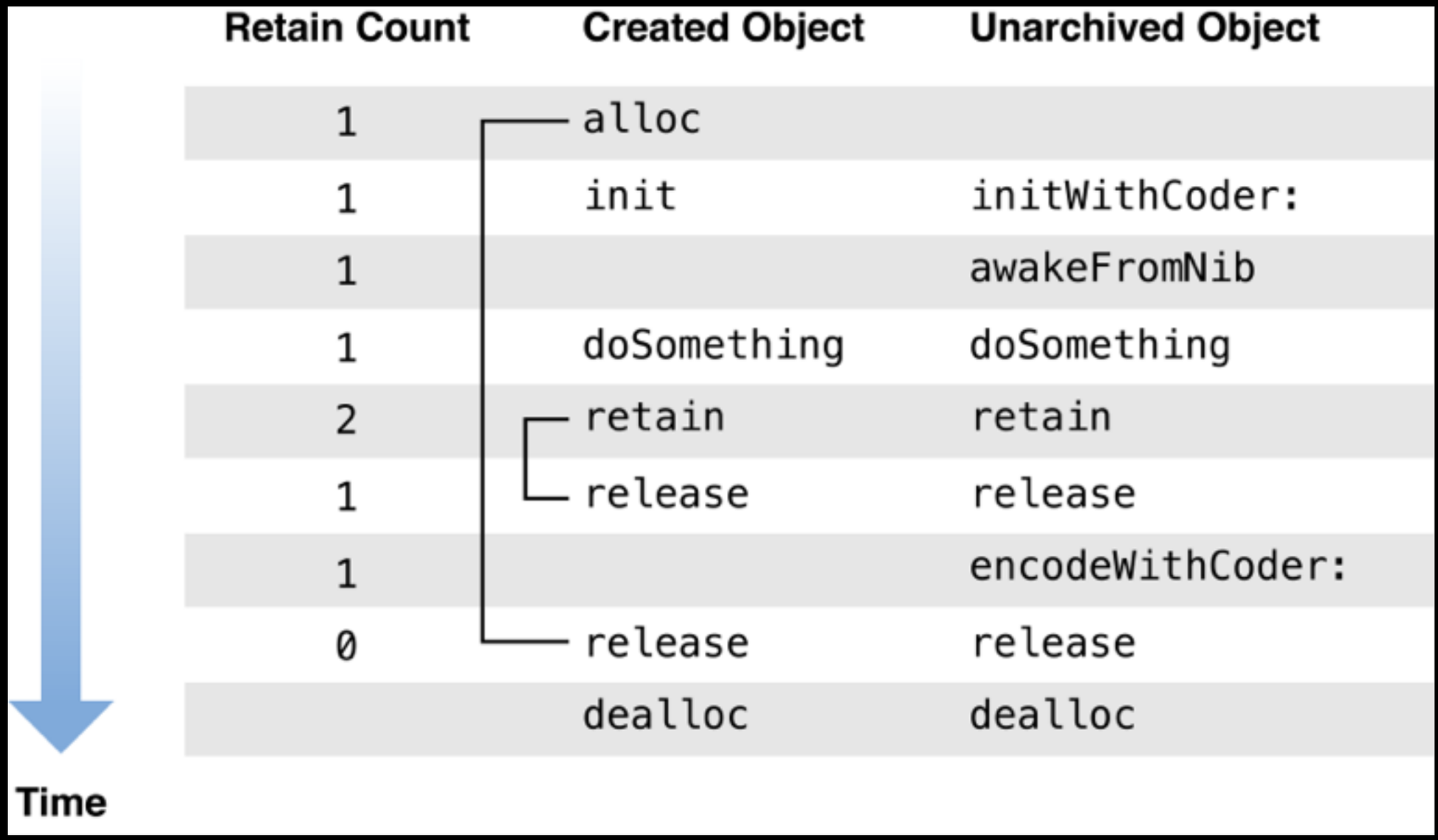
adopting protocol

```
//  BJObjCTestViewController.h

#import <UIKit/UIKit.h>

#import "BJWebSocketClient.h"

@interface BJObjCTestViewController : UIViewController <
BJWebSocketClientDelegate,
UIScrollViewDelegate>

@end
```

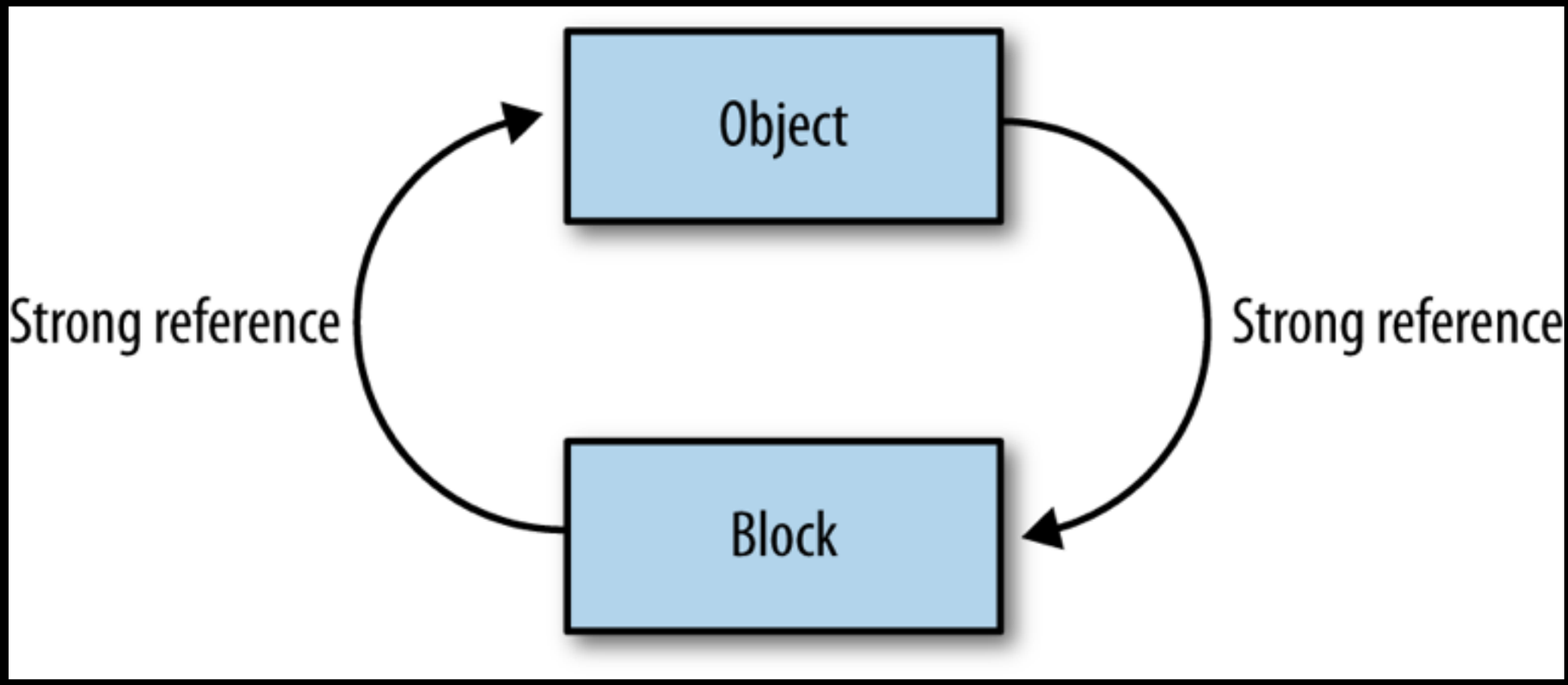
adopting protocol

```m
//  BJObjCTestViewController.m

#import "BJObjCTestViewController.h"

@implementation BJObjCTestViewController

//  ...

#pragma mark - <BJWebSocketClientDelegate>

- (void)webSocketClient:(BJWebSocketClient *)webSocketClient
          stateDidChange:(BJWSCState)state {
    NSLog(@"%@ state did change: %td", webSocketClient, state);
}

@end
```

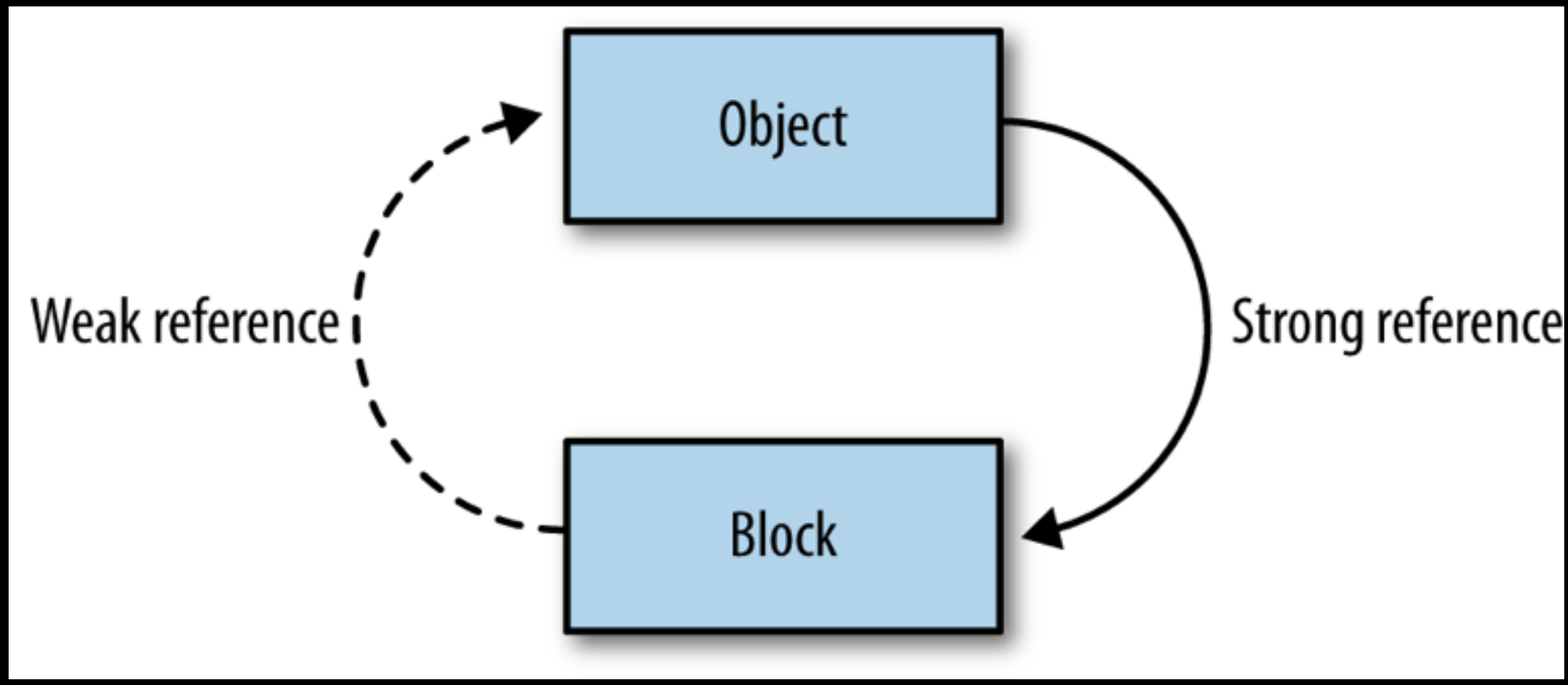
object



object life cycle



retain cycle



break retain cycle

method

method

```
- (void)webSocketClient:(BJWebSocketClient *)webSocketClient
          stateDidChange:(BJWSCState)state {
    // ...
}

[self.delegate webSocketClient:self didSendMessage:message];
```

selector

```objc
SEL selector = @selector(webSocketClient:didSendMessage:);

if ([self.delegate respondsToSelector:selector]) {
    [self.delegate webSocketClient:self didSendMessage:message];
}
```

super

```
- (void)webSocketClient:(BJWebSocketClient *)webSocketClient
        stateDidChange:(BJWSCState)state {
    if ([super respondsToSelector:_cmd]) {
        [self webSocketClient:webSocketClient stateDidChange:state];
    }
    NSLog(@"%@ state did change: %td", webSocketClient, state);
}
```

super

```
- (void)webSocketClient:(BJWebSocketClient *)webSocketClient
        stateDidChange:(BJWSCState)state {
    if ([self.superclass instancesRespondToSelector:_cmd]) {
        [self webSocketClient:webSocketClient stateDidChange:state];
    }
    NSLog(@"%@ state did change: %td", webSocketClient, state);
}
```

init

```
- (instancetype)init {
    self = [super init];
    if (self) {
        self.state = BJWSCState_connecting;
        _state = BJWSCState_connecting;
    }
    return self;
}
```

designated initializer

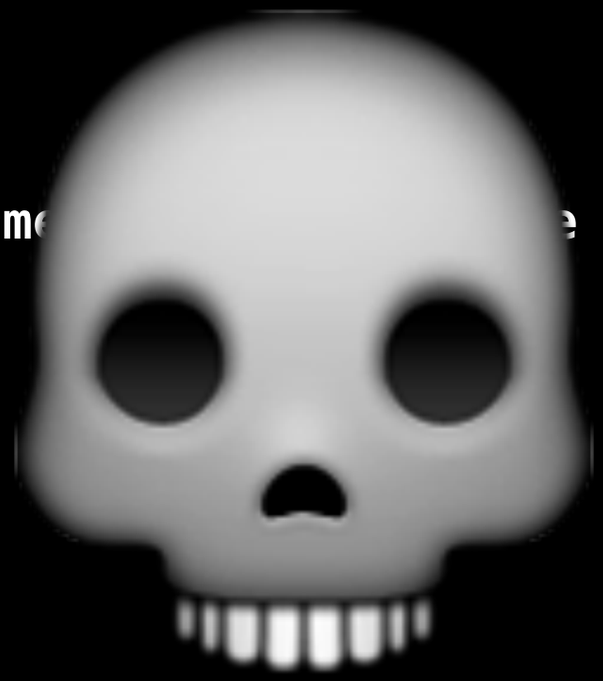
```
//  superclass.h

- (instancetype)init;
- (instancetype)initWithName:(NSString *)name NS_DESIGNATED_INITIALIZER;
```

designated initializer

```m
//  superclass.m

- (instancetype)init {
    return [self initWithName:nil];
}

- (instancetype)initWithName:(NSString *)name {
    self = [super init];
    if (self) {
        _name = name;
    }
    return self;
}
```

designated initializer

```
//  subclass.m

- (instancetype)initWithNam                    e {
    self = [super init];
    if (self) {
        _name = name;
        _nikename = name;
    }
    return self;
}
```



designated initializer

```
//  subclass.m
- (instancetype)initWithName:(NSString *)name {
    self = [super initWithName:name];
    if (self) {
        _nikename = name;
    }
    return self;
}
```

dealloc

```
- (void)dealloc {
    [[NSNotificationCenter defaultCenter] removeObserver:self];

    self.tableView.dataSource = nil;
    self.tableView.delegate = nil;
}
```

ivar

ivar

```
// @interface
@interface BJWebSocketClient : NSObject {
    BJWSCState _state;
}

@end
```

ivar

```
// @interface of class extension

@interface BJWebSocketClient () {
    BJWSCState _state;
}

@end
```

ivar

```
// @implementation
@implementation BJWebSocketClient {
    BJWSCState _state;
}

@end
```

protected

```
// @interface
@interface BJWebSocketClient : NSObject {
@protected // @private, @public
    BJWSCState _state;
}

@end
```

property

property

```
@property (nonatomic) NSURL *serverURL;
@property (nonatomic, copy) NSString *serverURLString;

@property (nonatomic, readonly) BJWSCState state;
@property (nonatomic, readwrite) BJWSCState state;

@property (nonatomic, copy) BJWSCStateCallback stateCallback;
@property (nonatomic, weak) id<BJWebSocketClientDelegate> delegate;
```

property

```
@property id x;
```

property

```
@property (atomic, readwrite, strong, getter=x, setter=setX:) id x;
```

property

```
@property (
    atomic,           // atomic, nonatomic
    readwrite,        // readonly, readwrite
    strong / assign, // assign, weak, strong, copy
    getter=name,      // getter=isSelected for BOOL selected
    setter=setName:   // setter=setSelected: for BOOL isSelected
) type name;
```

atomic vs nonatomic

```
@property (atomic) BJWSCState state;

@property (nonatomic) BJWSCState state;
```

readonly vs readwrite

```
@property (nonatomic, readonly) BJWSCState state;

@property (nonatomic, readwrite) BJWSCState state;
```

strong vs weak

```objc
@property (nonatomic, strong) NSURL *serverURL;

// weak + id<Protocol>
@property (nonatomic, weak) id<BJWebSocketClientDelegate> delegate;
```

assign vs weak

```objc
@property (nonatomic, assign) BJWSCState state;

@property (nonatomic, weak) id<BJWebSocketClientDelegate> delegate;
```

strong vs copy

```
@property (nonatomic, strong) NSURL *serverURL;

@property (nonatomic, copy) NSString *serverURLString;

@property (nonatomic, copy) NSArray<NSURL *> *serverURLs;

@property (nonatomic, copy) BJWSCStateCallback stateCallback;

@property (nonatomic, strong) BJWebSocketClient *webSocketClient;
```

copy & mutableCopy

```
[object copy];

[object mutableCopy];
```

NSCopying & NSMutableCopying

```
@protocol NSCopying

- (id)copyWithZone:(nullable NSZone *)zone;

@end

@protocol NSMutableCopying

- (id)mutableCopyWithZone:(nullable NSZone *)zone;

@end
```

immutable + mutable

- `[immutableObject copy]`

```
return self; // MRC: [self retain];
```

- `[immutableObject mutableCopy]`

```
BJMutableCopyable *copy = [[BJMutableCopyable alloc] init];
copy.prop = self.prop;
return copy;
```

- `[mutableObject copy]`

```
BJCopyable *copy = [[BJCopyable alloc] initWithProp:self.prop];
return copy;
```

- `[mutableObject mutableCopy]`

```
BJMutableCopyable *copy = [[BJMutableCopyable alloc] init];
copy.prop = self.prop;
return copy;
```

mutable-only

```objc
@interface BJMutableOnlyCopyable : NSObject <NSCopying>

@property (nonatomic, readwrite) id prop;

@end
```

```objc
@implementation BJMutableOnlyCopyable

#pragma mark - <NSCopying>

- (id)copyWithZone:(NSZone *)zone {
    BJMutableOnlyCopyable *copy = [[BJMutableOnlyCopyable alloc] init];
    copy.prop = self.prop;
    return copy;
}

@end
```

why

```
// immutable-only
@property (nonatomic, strong) NSURL *serverURL;

@property (nonatomic, copy) NSString *serverURLString;

@property (nonatomic, copy) NSArray<NSURL *> *serverURLs;

@property (nonatomic, copy) BJWSCStateCallback stateCallback;

// mutable-only
@property (nonatomic, strong) BJWebSocketClient *webSocketClient;
```

getter & setter

```objc
- (id)prop {
    return _prop;
}

// assign
- (void)setProp:(id)prop {
    _prop = prop;
}
```

```objc
// strong
- (void)setProp:(id)prop {
    [prop retain];   // 1
    [_prop release]; // 2
    _prop = prop;    // 3
}

// copy
- (void)setProp:(id)prop {
    prop = [prop copy]; // 1
    [_prop release];    // 2
    _prop = prop;       // 3
}
```

synthesize & dynamic

- empty
- synthesize

```
@synthesize prop = _prop;
```

- dynamic

```
@dynamic prop;

id _prop;

- (id)prop {
    return _prop;
}

- (void)setProp:(id)prop {
   _prop = prop;
}
```

block

retain cycle

```objc
@implementation BJObjCTestViewController

- (void)initialize {
    self.webSocketClient = [self makeWebSocketClient];
}

- (BJWebSocketClient *)makeWebSocketClient {
    BJWebSocketClient *client = [BJWebSocketClient new];
    client.stateCallback = ^(BJWSCState state) {
        [self printState:state];
    };
    return client;
}

@end
```

retain cycle

```objc
@implementation BJObjCTestViewController

- (void)initialize {
    self.webSocketClient = [self makeWebSocketClient];
}

- (BJWebSocketClient *)makeWebSocketClient {
    BJWebSocketClient *client = [BJWebSocketClient new];
    @weakify(self);
    client.stateCallback = ^(BJWSCState state) {
        @strongify(self);
        [self printState:state];
    };
    return client;
}

@end
```

retain cycle

```
- (BJWebSocketClient *)makeWebSocketClient {
    BJWebSocketClient *client = [BJWebSocketClient new];
    @weakify(self);
    client.stateCallback = ^(BJWSCState state) {
        @strongify(self);
        [self printState:client.state];
    };
    return client;
}
```

retain cycle

```
- (BJWebSocketClient *)makeWebSocketClient {
    BJWebSocketClient *client = [BJWebSocketClient new];
    @weakify(self, client);
    client.stateCallback = ^(BJWSCState state) {
        @strongify(self, client);
        [self printState:client.state];
    };
    return client;
}
```

retain cycle

```
- (BJWebSocketClient *)makeWebSocketClient {
    BJWebSocketClient *client = [BJWebSocketClient new];
    client.stateCallback = ^(BJWSCState state) {
        NSLog(@"state did change: %td”, _webSocketClient.state);
    };
    return client;
}
```

retain cycle

```objc
- (BJWebSocketClient *)makeWebSocketClient {
    BJWebSocketClient *client = [BJWebSocketClient new];
    @weakify(self);
    client.stateCallback = ^(BJWSCState state) {
        @strongify(self);
        NSLog(@"state did change: %td", self->_webSocketClient.state);
    };
    return client;
}
```

__weak & __strong

```
@weakify(self):
    typeof(self) __weak __weak_self__ = self;

@strongify(self):
    typeof(self) __strong self = __weak_self__;
```

__weak & __strong

```
- (BJWebSocketClient *)makeWebSocketClient {
    BJWebSocketClient *client = [BJWebSocketClient new];
    typeof(self) __weak __weak_self__ = self;
    client.stateCallback = ^(BJWSCState state) {
        typeof(self) __strong self = __weak_self__;
        if (!self) {
            return;
        }
        [aMutableArray addObject:self];
        [self printState:state];
    };
    return client;
}
```

__weak & __strong

```objc
- (BJWebSocketClient *)makeWebSocketClient {
    BJWebSocketClient *client = [BJWebSocketClient new];
    @weakify(self, client);
    client.stateCallback = ^(BJWSCState state) {
        @strongify(self, client);
        [self printState:state];

        @weakify(self); // ???:
        client.stateCallback = ^(BJWSCState state) {
            @strongify(self); // ???:
            [self printState:state];
        };
    };
    return client;
}
```

__weak & __strong

```
- (BJWebSocketClient *)makeWebSocketClient {
    BJWebSocketClient *client = [BJWebSocketClient new];
    @weakify(self);
    client.stateCallback = ^(BJWSCState state) {
        @strongify(self);
        [self printState:state];

        @weakify(self);
        client.stateCallback = ^(BJWSCState state) {
            @strongify(self); // √
            [self printState:state];
        };
    };
    return client;
}
```

__block

```
NSInteger __block total = 0;
void (^plus)(NSInteger x) = ^void (NSInteger x) {
    total += x;
};

for (NSInteger i = 1; i <= 10; i++) {
    plus(i);
}
```

Cocoa Touch

- **Foundation**
- **UIKit**
- . . .

Documentation!

- ⌘⇧0

  Documentation and API Reference

- ^⌥⌘/

  Search Documentation for Selected Text

git

git

http://git.baijiahulian.com/iOS/training

one more thing…

homework

- NSNotificationCenter

NSNotificationCenter

```
@interface NSNotificationCenter : NSObject

+ (NSNotificationCenter *)defaultCenter;

- (void)postNotification:(NSNotification *)notification;
- (void)postNotificationName:(NSString *)aName
                      object:(nullable id)anObject;
- (void)postNotificationName:(NSString *)aName
                      object:(nullable id)anObject
                    userInfo:(nullable NSDictionary *)aUserInfo;
```

NSNotificationCenter

```
- (void)addObserver:(id)observer
           selector:(SEL)aSelector
               name:(nullable NSString *)aName
             object:(nullable id)anObject;
- (id <NSObject>)addObserverForName:(nullable NSString *)name
                             object:(nullable id)obj
                              queue:(nullable NSOperationQueue *)queue
                         usingBlock:(void (^)(NSNotification *n))block;

- (void)removeObserver:(id)observer;
- (void)removeObserver:(id)observer
                  name:(nullable NSString *)aName
                object:(nullable id)anObject;

@end
```

homework

minglinqing@baijiahulian.com

Earlier than Next Tuesday

the end