iOS:

//

// AllChatBots.h

// AIChatBots

//

// Created by yangboz on 2016/12/26.

// Copyright © 2016年 \_\_\_SMARTKIT.INFO\_\_\_. All rights reserved.

//

#import "JSONModel.h"

#import "ChatBotVoModel.h"

@protocol ChatBotVoModel

@end

@interface AllChatBotsModel:JSONModel

@property(nonatomic) NSArray <ChatBotVoModel> \*chatbots;

@end

//

// APIFullMessageVoModel.h

// AIChatBots

//

// Created by yangboz on 2016/12/28.

// Copyright © 2016年 \_\_\_SMARTKIT.INFO\_\_\_. All rights reserved.

//

#import <Foundation/Foundation.h>

#import "JSONModel.h"

#import "APIUserVoModel.h"

#import "APIMessageVoModel.h"

@protocol APIUserVoModel

@end

@protocol APIMessageVoModel

@end

@interface APIFullMessageVoModel : JSONModel

@property(nonatomic) APIMessageVoModel \*message;

@property(nonatomic) APIUserVoModel \*user;

@end

//

// APIMessageVoModel.h

// AIChatBots

//

// Created by yangboz on 2016/12/28.

// Copyright © 2016年 \_\_\_SMARTKIT.INFO\_\_\_. All rights reserved.

//

#import <Foundation/Foundation.h>

#import "JSONModel.h"

@interface APIMessageVoModel : JSONModel

@property(nonatomic) NSString \*message;

@property(nonatomic) NSNumber \*chatBotID;

@property(nonatomic) NSNumber \*timestamp;

@end

//

// APIUserVoModel.h

// AIChatBots

//

// Created by yangboz on 2016/12/28.

// Copyright © 2016年 \_\_\_SMARTKIT.INFO\_\_\_. All rights reserved.

//

#import <Foundation/Foundation.h>

#import "JSONModel.h"

@interface APIUserVoModel : JSONModel

@property(nonatomic) NSString \*firstName;

@property(nonatomic) NSString \*lastName;

@property(nonatomic) NSString \*gender;

@property(nonatomic) NSString \*externalID;

@end

//

// AppDelegate.h

// AIChatBots

//

// Created by yangboz on 2016/12/26.

// Copyright © 2016年 \_\_\_SMARTKIT.INFO\_\_\_. All rights reserved.

//

#import <UIKit/UIKit.h>

@interface AppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow \*window;

-(void)setMasterControllerData:(NSMutableArray \*)data;

@end

//

// ChatBotVo.h

// ChatBotsMessager

//

// Created by zhou Yangbo on 13-3-19.

// Copyright (c) 2013年 GODPAPER. All rights reserved.

//

#import "JSONModel.h"

@interface ChatBotVoModel:JSONModel

//"image": "prob\_23958.gif",

//"id": 23958,

//"name": "prob",

//"Development": 29715,

//"AI": "•••••••••",

//"Updated": "Cold",

//"Bio": "Problem is a wood elf, she loves to help",

//"Entity": "Mythical",

//"Personality": "Helper",

//"Temperament": "Friendly",

//"Basis": "Original",

//"From": "The Grove of Sumber, Crescent Mountains",

//"Country": "Middle Earth",

//"Gender": "F",

//"Created": "February 19, 2005",

// "Interests": "Prob is not an adult bot. PLEASE DO NOT CYBER Flora and fauna, singing, and roaming the forests are Prob's favorite things to do. She also has a fascination with campfires email her at Problem@WitchesBrew.zzn.com This is NOT an adult bot.. "

//,"Rating": "E"

@property(nonatomic) NSString \*Image;

@property(nonatomic) NSNumber \*Id;

@property(nonatomic) NSString \*Name;

@property(nonatomic) NSNumber \*Development;

@property(nonatomic) NSString \*AI;

@property(nonatomic) NSString \*Updated;

@property(nonatomic) NSString \*Bio;

@property(nonatomic) NSString \*Entity;

@property(nonatomic) NSString \*Personality;

@property(nonatomic) NSString \*Temperament;

@property(nonatomic) NSString \*Basis;

@property(nonatomic) NSString \*From;

@property(nonatomic) NSString \*Country;

@property(nonatomic) NSString \*Gender;

@property(nonatomic) NSString \*Created;

@property(nonatomic) NSString \*Interests;

@property(nonatomic) NSString \*Rating;

@end

//

// Constants.h

// ChatBotsMessager

//

// Created by zhou Yangbo on 13-3-19.

// Copyright (c) 2013年 GODPAPER. All rights reserved.

//

#import <Foundation/Foundation.h>

@interface Constants : NSObject

FOUNDATION\_EXTERN NSString \*const API\_DOMAIN;

FOUNDATION\_EXTERN NSString \*const API\_KEY;

FOUNDATION\_EXTERN NSString \*const API\_SECERT;

FOUNDATION\_EXTERN NSString \*const API\_HASH;

@end

//

// COVITASv1API.h

// DeepDetectChatBots

//

// Created by yangboz on 24/03/2017.

// Copyright © 2017 \_\_\_SMARTKIT.INFO\_\_\_. All rights reserved.

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

#import "MessageVoModel.h"

#import "COVITASv1Model.h"

#import "MessageVoSimple.h"

//@see: http://stackoverflow.com/questions/5643514/how-to-define-an-nsstring-for-global-use

//#define DEV @"dev\_aliyun"

#ifdef DEV

#define kAPIEndpointHost @"http://118.190.3.169:8085/covitas/tas"

#else//LOCAL

#define kAPIEndpointHost @"http://192.168.0.11:8085/covitas/tas"

#endif

#define kAPI\_chat\_pf (@"pf")//pf/{chatBotID}/{message}

//Notification Center post names;

#define kNCpN\_chat\_with\_message @"pfChatWithMessage"

#define kNCpN\_chat\_with\_message\_error @"pfChatWithMessageError"

//Default value.

#define kAPI\_default\_chatbot\_id 6

#define kAPI\_default\_chatbot\_ex\_id @"abc-639184572"

#define kAPI\_default\_chatbot\_fname @"Tugger"

#define kAPI\_default\_chatbot\_lname @"Sufani"

#define kAPI\_default\_chatbot\_gender @"m"

@interface COVITASv1API : NSObject

+(COVITASv1API \*)sharedInstance;

//@see:https://www.personalityforge.com/chatbot-api-docs.php

-(void)pfChatWithGetMessage:(MessageVoSimple\*)vo;

-(void)pfChatWithPostMessage:(MessageVoSimple\*)vo;

@end

//

// COVIASv1Model.h

// DeepDetectChatBots

//

// Created by yangboz on 24/03/2017.

// Copyright © 2017 \_\_\_SMARTKIT.INFO\_\_\_. All rights reserved.

//

#import <Foundation/Foundation.h>

#import "MessageVoModel.h"

@interface COVITASv1Model : NSObject

+(COVITASv1Model \*)sharedInstance;

@property (strong, nonatomic) MessageVoModel \*messageVo;

@end

//

// DataModel.h

// TGCameraViewController

//

// Created by yangboz on 2016/12/1.

// Copyright © 2016年 Tudo Gostoso Internet. All rights reserved.

//

#ifndef DataModel\_h

#define DataModel\_h

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

#import "AllChatBotsModel.h"

@interface DataModel : NSObject

{

}

+(DataModel \*)sharedInstance;

//Set/Gets

-(void)setAllChatBots:(AllChatBotsModel \*)value;

-(AllChatBotsModel \*)getAllChatBots;

//Current selected chat bot

-(ChatBotVoModel \*)getSelectedChatBot;

-(void)setSelectedChatBot:(ChatBotVoModel \*)value;

@end

#endif /\* DataModel\_h \*/

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

// Import all the things

#import "JSQMessages.h"

#import "DemoModelData.h"

#import "NSUserDefaults+DemoSettings.h"

#import "ChatBotVoModel.h"

@class DemoMessagesViewController;

@protocol JSQDemoViewControllerDelegate <NSObject>

- (void)didDismissJSQDemoViewController:(DemoMessagesViewController \*)vc;

@end

@interface DemoMessagesViewController : JSQMessagesViewController <UISplitViewControllerDelegate,UIActionSheetDelegate, JSQMessagesComposerTextViewPasteDelegate>

@property (weak, nonatomic) id<JSQDemoViewControllerDelegate> delegateModal;

@property (strong, nonatomic) ChatBotVoModel \*detailItem;

@property (strong, nonatomic) DemoModelData \*demoData;

- (void)receiveMessagePressed:(UIBarButtonItem \*)sender;

- (void)closePressed:(UIBarButtonItem \*)sender;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

#import <CoreLocation/CoreLocation.h>

#import "JSQMessages.h"

/\*\*

\* This is for demo/testing purposes only.

\* This object sets up some fake model data.

\* Do not actually do anything like this.

\*/

static NSString \* const kJSQDemoAvatarDisplayNameSquires = @"Jesse Squires";

static NSString \* const kJSQDemoAvatarDisplayNameCook = @"Tim Cook";

static NSString \* const kJSQDemoAvatarDisplayNameJobs = @"Jobs";

static NSString \* const kJSQDemoAvatarDisplayNameWoz = @"Steve Wozniak";

static NSString \* const kJSQDemoAvatarIdSquires = @"053496-4509-289";

static NSString \* const kJSQDemoAvatarIdCook = @"468-768355-23123";

static NSString \* const kJSQDemoAvatarIdJobs = @"707-8956784-57";

static NSString \* const kJSQDemoAvatarIdWoz = @"309-41802-93823";

@interface DemoModelData : NSObject

@property (strong, nonatomic) NSMutableArray \*messages;

@property (strong, nonatomic) NSDictionary \*avatars;

@property (strong, nonatomic) JSQMessagesBubbleImage \*outgoingBubbleImageData;

@property (strong, nonatomic) JSQMessagesBubbleImage \*incomingBubbleImageData;

@property (strong, nonatomic) NSDictionary \*users;

- (void)addPhotoMediaMessage;

- (void)addLocationMediaMessageCompletion:(JSQLocationMediaItemCompletionBlock)completion;

- (void)addVideoMediaMessage;

- (void)addVideoMediaMessageWithThumbnail;

- (void)addAudioMediaMessage;

@end

//

// DetailViewController.h

// AIChatBots

//

// Created by yangboz on 2016/12/26.

// Copyright © 2016年 \_\_\_SMARTKIT.INFO\_\_\_. All rights reserved.

//

#import <UIKit/UIKit.h>

#import "JSQMessagesViewController.h"

#import "ChatBotVoModel.h"

@interface DetailViewController : UIViewController

@property (strong, nonatomic) ChatBotVoModel \*detailItem;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <JSQSystemSoundPlayer/JSQSystemSoundPlayer.h>

@interface JSQSystemSoundPlayer (JSQMessages)

/\*\*

\* Plays the default sound for received messages.

\*/

+ (void)jsq\_playMessageReceivedSound;

/\*\*

\* Plays the default sound for received messages \*as an alert\*, invoking device vibration if available.

\*/

+ (void)jsq\_playMessageReceivedAlert;

/\*\*

\* Plays the default sound for sent messages.

\*/

+ (void)jsq\_playMessageSentSound;

/\*\*

\* Plays the default sound for sent messages \*as an alert\*, invoking device vibration if available.

\*/

+ (void)jsq\_playMessageSentAlert;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

@interface NSBundle (JSQMessages)

/\*\*

\* @return The bundle for JSQMessagesViewController.

\*/

+ (NSBundle \*)jsq\_messagesBundle;

/\*\*

\* @return The bundle for assets in JSQMessagesViewController.

\*/

+ (NSBundle \*)jsq\_messagesAssetBundle;

/\*\*

\* Returns a localized version of the string designated by the specified key and residing in the JSQMessages table.

\*

\* @param key The key for a string in the JSQMessages table.

\*

\* @return A localized version of the string designated by key in the JSQMessages table.

\*/

+ (NSString \*)jsq\_localizedStringForKey:(NSString \*)key;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

@interface NSString (JSQMessages)

/\*\*

\* @return A copy of the receiver with all leading and trailing whitespace removed.

\*/

- (NSString \*)jsq\_stringByTrimingWhitespace;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <UIKit/UIKit.h>

@interface UIColor (JSQMessages)

#pragma mark - Message bubble colors

/\*\*

\* @return A color object containing HSB values similar to the iOS 7 messages app green bubble color.

\*/

+ (UIColor \*)jsq\_messageBubbleGreenColor;

/\*\*

\* @return A color object containing HSB values similar to the iOS 7 messages app blue bubble color.

\*/

+ (UIColor \*)jsq\_messageBubbleBlueColor;

/\*\*

\* @return A color object containing HSB values similar to the iOS 7 red color.

\*/

+ (UIColor \*)jsq\_messageBubbleRedColor;

/\*\*

\* @return A color object containing HSB values similar to the iOS 7 messages app light gray bubble color.

\*/

+ (UIColor \*)jsq\_messageBubbleLightGrayColor;

#pragma mark - Utilities

/\*\*

\* Creates and returns a new color object whose brightness component is decreased by the given value, using the initial color values of the receiver.

\*

\* @param value A floating point value describing the amount by which to decrease the brightness of the receiver.

\*

\* @return A new color object whose brightness is decreased by the given values. The other color values remain the same as the receiver.

\*/

- (UIColor \*)jsq\_colorByDarkeningColorWithValue:(CGFloat)value;

@end//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <UIKit/UIKit.h>

@interface UIDevice (JSQMessages)

/\*\*

\* @return Whether or not the current device is running a version of iOS before 8.0.

\*/

+ (BOOL)jsq\_isCurrentDeviceBeforeiOS8;

/\*\*

\* @return Whether or not the current device is running a version of iOS after 9.0.

\*/

+ (BOOL)jsq\_isCurrentDeviceAfteriOS9;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <UIKit/UIKit.h>

@interface UIImage (JSQMessages)

/\*\*

\* Creates and returns a new image object that is masked with the specified mask color.

\*

\* @param maskColor The color value for the mask. This value must not be `nil`.

\*

\* @return A new image object masked with the specified color.

\*/

- (UIImage \*)jsq\_imageMaskedWithColor:(UIColor \*)maskColor;

/\*\*

\* @return The regular message bubble image.

\*/

+ (UIImage \*)jsq\_bubbleRegularImage;

/\*\*

\* @return The regular message bubble image without a tail.

\*/

+ (UIImage \*)jsq\_bubbleRegularTaillessImage;

/\*\*

\* @return The regular message bubble image stroked, not filled.

\*/

+ (UIImage \*)jsq\_bubbleRegularStrokedImage;

/\*\*

\* @return The regular message bubble image stroked, not filled and without a tail.

\*/

+ (UIImage \*)jsq\_bubbleRegularStrokedTaillessImage;

/\*\*

\* @return The compact message bubble image.

\*

\* @discussion This is the default bubble image used by `JSQMessagesBubbleImageFactory`.

\*/

+ (UIImage \*)jsq\_bubbleCompactImage;

/\*\*

\* @return The compact message bubble image without a tail.

\*/

+ (UIImage \*)jsq\_bubbleCompactTaillessImage;

/\*\*

\* @return The default input toolbar accessory image.

\*/

+ (UIImage \*)jsq\_defaultAccessoryImage;

/\*\*

\* @return The default typing indicator image.

\*/

+ (UIImage \*)jsq\_defaultTypingIndicatorImage;

/\*\*

\* @return The default play icon image.

\*/

+ (UIImage \*)jsq\_defaultPlayImage;

/\*\*

\* @return The default pause icon image.

\*/

+ (UIImage \*)jsq\_defaultPauseImage;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <UIKit/UIKit.h>

@interface UIView (JSQMessages)

/\*\*

\* Pins the subview of the receiver to the edge of its frame, as specified by the given attribute, by adding a layout constraint.

\*

\* @param subview The subview to which the receiver will be pinned.

\* @param attribute The layout constraint attribute specifying one of `NSLayoutAttributeBottom`, `NSLayoutAttributeTop`, `NSLayoutAttributeLeading`, `NSLayoutAttributeTrailing`.

\*/

- (void)jsq\_pinSubview:(UIView \*)subview toEdge:(NSLayoutAttribute)attribute;

/\*\*

\* Pins all edges of the specified subview to the receiver.

\*

\* @param subview The subview to which the receiver will be pinned.

\*/

- (void)jsq\_pinAllEdgesOfSubview:(UIView \*)subview;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

//

// Ideas for keyboard controller taken from Daniel Amitay

// DAKeyboardControl

// https://github.com/danielamitay/DAKeyboardControl

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

@class JSQMessagesKeyboardController;

/\*\*

\* Posted when the system keyboard frame changes.

\* The object of the notification is the `JSQMessagesKeyboardController` object.

\* The `userInfo` dictionary contains the new keyboard frame for key

\* `JSQMessagesKeyboardControllerUserInfoKeyKeyboardDidChangeFrame`.

\*/

FOUNDATION\_EXPORT NSString \* const JSQMessagesKeyboardControllerNotificationKeyboardDidChangeFrame;

/\*\*

\* Contains the new keyboard frame wrapped in an `NSValue` object.

\*/

FOUNDATION\_EXPORT NSString \* const JSQMessagesKeyboardControllerUserInfoKeyKeyboardDidChangeFrame;

/\*\*

\* The `JSQMessagesKeyboardControllerDelegate` protocol defines methods that

\* allow you to respond to the frame change events of the system keyboard.

\*

\* A `JSQMessagesKeyboardController` object also posts the `JSQMessagesKeyboardControllerNotificationKeyboardDidChangeFrame`

\* in response to frame change events of the system keyboard.

\*/

@protocol JSQMessagesKeyboardControllerDelegate <NSObject>

@required

/\*\*

\* Tells the delegate that the keyboard frame has changed.

\*

\* @param keyboardController The keyboard controller that is notifying the delegate.

\* @param keyboardFrame The new frame of the keyboard in the coordinate system of the `contextView`.

\*/

- (void)keyboardController:(JSQMessagesKeyboardController \*)keyboardController keyboardDidChangeFrame:(CGRect)keyboardFrame;

@end

/\*\*

\* An instance of `JSQMessagesKeyboardController` manages responding to the hiding and showing

\* of the system keyboard for editing its `textView` within its specified `contextView`.

\* It also controls user interaction with the system keyboard via its `panGestureRecognizer`,

\* allow the user to interactively pan the keyboard up and down in the `contextView`.

\*

\* When the system keyboard frame changes, it posts the `JSQMessagesKeyboardControllerNotificationKeyboardDidChangeFrame`.

\*/

@interface JSQMessagesKeyboardController : NSObject

/\*\*

\* The object that acts as the delegate of the keyboard controller.

\*/

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

/\*\*

\* The text view in which the user is editing with the system keyboard.

\*/

@property (weak, nonatomic, readonly) UITextView \*textView;

/\*\*

\* The view in which the keyboard will be shown. This should be the parent or a sibling of `textView`.

\*/

@property (weak, nonatomic, readonly) UIView \*contextView;

/\*\*

\* The pan gesture recognizer responsible for handling user interaction with the system keyboard.

\*/

@property (weak, nonatomic, readonly) UIPanGestureRecognizer \*panGestureRecognizer;

/\*\*

\* Specifies the distance from the keyboard at which the `panGestureRecognizer`

\* should trigger user interaction with the keyboard by panning.

\*

\* @discussion The x value of the point is not used.

\*/

@property (assign, nonatomic) CGPoint keyboardTriggerPoint;

/\*\*

\* Returns `YES` if the keyboard is currently visible, `NO` otherwise.

\*/

@property (assign, nonatomic, readonly) BOOL keyboardIsVisible;

/\*\*

\* Returns the current frame of the keyboard if it is visible, otherwise `CGRectNull`.

\*/

@property (assign, nonatomic, readonly) CGRect currentKeyboardFrame;

/\*\*

\* Not a valid initializer.

\*/

- (id)init NS\_UNAVAILABLE;

/\*\*

\* Creates a new keyboard controller object with the specified textView, contextView, panGestureRecognizer, and delegate.

\*

\* @param textView The text view in which the user is editing with the system keyboard. This value must not be `nil`.

\* @param contextView The view in which the keyboard will be shown. This should be the parent or a sibling of `textView`. This value must not be `nil`.

\* @param panGestureRecognizer The pan gesture recognizer responsible for handling user interaction with the system keyboard. This value must not be `nil`.

\* @param delegate The object that acts as the delegate of the keyboard controller.

\*

\* @return An initialized `JSQMessagesKeyboardController` if created successfully, `nil` otherwise.

\*/

- (instancetype)initWithTextView:(UITextView \*)textView

contextView:(UIView \*)contextView

panGestureRecognizer:(UIPanGestureRecognizer \*)panGestureRecognizer

delegate:(id<JSQMessagesKeyboardControllerDelegate>)delegate NS\_DESIGNATED\_INITIALIZER;

/\*\*

\* Tells the keyboard controller that it should begin listening for system keyboard notifications.

\*/

- (void)beginListeningForKeyboard;

/\*\*

\* Tells the keyboard controller that it should end listening for system keyboard notifications.

\*/

- (void)endListeningForKeyboard;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <UIKit/UIKit.h>

#import "JSQMessagesCollectionView.h"

#import "JSQMessagesCollectionViewFlowLayout.h"

#import "JSQMessagesInputToolbar.h"

#import "JSQMessagesKeyboardController.h"

/\*\*

\* The `JSQMessagesViewController` class is an abstract class that represents a view controller whose content consists of

\* a `JSQMessagesCollectionView` and `JSQMessagesInputToolbar` and is specialized to display a messaging interface.

\*

\* @warning This class is intended to be subclassed. You should not use it directly.

\*/

@interface JSQMessagesViewController : UIViewController <JSQMessagesCollectionViewDataSource,

JSQMessagesCollectionViewDelegateFlowLayout,

UITextViewDelegate>

/\*\*

\* Returns the collection view object managed by this view controller.

\* This view controller is the collection view's data source and delegate.

\*/

@property (weak, nonatomic, readonly) JSQMessagesCollectionView \*collectionView;

/\*\*

\* Returns the input toolbar view object managed by this view controller.

\* This view controller is the toolbar's delegate.

\*/

@property (weak, nonatomic, readonly) JSQMessagesInputToolbar \*inputToolbar;

/\*\*

\* Returns the keyboard controller object used to manage the software keyboard.

\*/

@property (strong, nonatomic) JSQMessagesKeyboardController \*keyboardController;

/\*\*

\* The display name of the current user who is sending messages.

\*

\* @discussion This value does not have to be unique. This value must not be `nil`.

\*/

@property (copy, nonatomic) NSString \*senderDisplayName;

/\*\*

\* The string identifier that uniquely identifies the current user sending messages.

\*

\* @discussion This property is used to determine if a message is incoming or outgoing.

\* All message data objects returned by `collectionView:messageDataForItemAtIndexPath:` are

\* checked against this identifier. This value must not be `nil`.

\*/

@property (copy, nonatomic) NSString \*senderId;

/\*\*

\* Specifies whether or not the view controller should automatically scroll to the most recent message

\* when the view appears and when sending, receiving, and composing a new message.

\*

\* @discussion The default value is `YES`, which allows the view controller to scroll automatically to the most recent message.

\* Set to `NO` if you want to manage scrolling yourself.

\*/

@property (assign, nonatomic) BOOL automaticallyScrollsToMostRecentMessage;

/\*\*

\* The collection view cell identifier to use for dequeuing outgoing message collection view cells

\* in the collectionView for text messages.

\*

\* @discussion This cell identifier is used for outgoing text message data items.

\* The default value is the string returned by `[JSQMessagesCollectionViewCellOutgoing cellReuseIdentifier]`.

\* This value must not be `nil`.

\*

\* @see JSQMessagesCollectionViewCellOutgoing.

\*

\* @warning Overriding this property's default value is \*not\* recommended.

\* You should only override this property's default value if you are proividing your own cell prototypes.

\* These prototypes must be registered with the collectionView for reuse and you are then responsible for

\* completely overriding many delegate and data source methods for the collectionView,

\* including `collectionView:cellForItemAtIndexPath:`.

\*/

@property (copy, nonatomic) NSString \*outgoingCellIdentifier;

/\*\*

\* The collection view cell identifier to use for dequeuing outgoing message collection view cells

\* in the collectionView for media messages.

\*

\* @discussion This cell identifier is used for outgoing media message data items.

\* The default value is the string returned by `[JSQMessagesCollectionViewCellOutgoing mediaCellReuseIdentifier]`.

\* This value must not be `nil`.

\*

\* @see JSQMessagesCollectionViewCellOutgoing.

\*

\* @warning Overriding this property's default value is \*not\* recommended.

\* You should only override this property's default value if you are proividing your own cell prototypes.

\* These prototypes must be registered with the collectionView for reuse and you are then responsible for

\* completely overriding many delegate and data source methods for the collectionView,

\* including `collectionView:cellForItemAtIndexPath:`.

\*/

@property (copy, nonatomic) NSString \*outgoingMediaCellIdentifier;

/\*\*

\* The collection view cell identifier to use for dequeuing incoming message collection view cells

\* in the collectionView for text messages.

\*

\* @discussion This cell identifier is used for incoming text message data items.

\* The default value is the string returned by `[JSQMessagesCollectionViewCellIncoming cellReuseIdentifier]`.

\* This value must not be `nil`.

\*

\* @see JSQMessagesCollectionViewCellIncoming.

\*

\* @warning Overriding this property's default value is \*not\* recommended.

\* You should only override this property's default value if you are proividing your own cell prototypes.

\* These prototypes must be registered with the collectionView for reuse and you are then responsible for

\* completely overriding many delegate and data source methods for the collectionView,

\* including `collectionView:cellForItemAtIndexPath:`.

\*/

@property (copy, nonatomic) NSString \*incomingCellIdentifier;

/\*\*

\* The collection view cell identifier to use for dequeuing incoming message collection view cells

\* in the collectionView for media messages.

\*

\* @discussion This cell identifier is used for incoming media message data items.

\* The default value is the string returned by `[JSQMessagesCollectionViewCellIncoming mediaCellReuseIdentifier]`.

\* This value must not be `nil`.

\*

\* @see JSQMessagesCollectionViewCellIncoming.

\*

\* @warning Overriding this property's default value is \*not\* recommended.

\* You should only override this property's default value if you are proividing your own cell prototypes.

\* These prototypes must be registered with the collectionView for reuse and you are then responsible for

\* completely overriding many delegate and data source methods for the collectionView,

\* including `collectionView:cellForItemAtIndexPath:`.

\*/

@property (copy, nonatomic) NSString \*incomingMediaCellIdentifier;

/\*\*

\* Specifies whether or not the view controller should show the typing indicator for an incoming message.

\*

\* @discussion Setting this property to `YES` will animate showing the typing indicator immediately.

\* Setting this property to `NO` will animate hiding the typing indicator immediately. You will need to scroll

\* to the bottom of the collection view in order to see the typing indicator. You may use `scrollToBottomAnimated:` for this.

\*/

@property (assign, nonatomic) BOOL showTypingIndicator;

/\*\*

\* Specifies whether or not the view controller should show the "load earlier messages" header view.

\*

\* @discussion Setting this property to `YES` will show the header view immediately.

\* Settings this property to `NO` will hide the header view immediately. You will need to scroll to

\* the top of the collection view in order to see the header.

\*/

@property (assign, nonatomic) BOOL showLoadEarlierMessagesHeader;

/\*\*

\* Specifies an additional inset amount to be added to the collectionView's contentInsets.top value.

\*

\* @discussion Use this property to adjust the top content inset to account for a custom subview at the top of your view controller.

\*/

@property (assign, nonatomic) CGFloat topContentAdditionalInset;

#pragma mark - Class methods

/\*\*

\* Returns the `UINib` object initialized for a `JSQMessagesViewController`.

\*

\* @return The initialized `UINib` object or `nil` if there were errors during initialization

\* or the nib file could not be located.

\*

\* @discussion You may override this method to provide a customized nib. If you do,

\* you should also override `messagesViewController` to return your

\* view controller loaded from your custom nib.

\*/

+ (UINib \*)nib;

/\*\*

\* Creates and returns a new `JSQMessagesViewController` object.

\*

\* @discussion This is the designated initializer for programmatic instantiation.

\*

\* @return An initialized `JSQMessagesViewController` object if successful, `nil` otherwise.

\*/

+ (instancetype)messagesViewController;

#pragma mark - Messages view controller

/\*\*

\* This method is called when the user taps the send button on the inputToolbar

\* after composing a message with the specified data.

\*

\* @param button The send button that was pressed by the user.

\* @param text The message text.

\* @param senderId The message sender identifier.

\* @param senderDisplayName The message sender display name.

\* @param date The message date.

\*/

- (void)didPressSendButton:(UIButton \*)button

withMessageText:(NSString \*)text

senderId:(NSString \*)senderId

senderDisplayName:(NSString \*)senderDisplayName

date:(NSDate \*)date;

/\*\*

\* This method is called when the user taps the accessory button on the `inputToolbar`.

\*

\* @param sender The accessory button that was pressed by the user.

\*/

- (void)didPressAccessoryButton:(UIButton \*)sender;

/\*\*

\* Animates the sending of a new message. See `finishSendingMessageAnimated:` for more details.

\*

\* @see `finishSendingMessageAnimated:`.

\*/

- (void)finishSendingMessage;

/\*\*

\* Completes the "sending" of a new message by resetting the `inputToolbar`, adding a new collection view cell in the collection view,

\* reloading the collection view, and scrolling to the newly sent message as specified by `automaticallyScrollsToMostRecentMessage`.

\* Scrolling to the new message can be animated as specified by the animated parameter.

\*

\* @param animated Specifies whether the sending of a message should be animated or not. Pass `YES` to animate changes, `NO` otherwise.

\*

\* @discussion You should call this method at the end of `didPressSendButton: withMessageText: senderId: senderDisplayName: date`

\* after adding the new message to your data source and performing any related tasks.

\*

\* @see `automaticallyScrollsToMostRecentMessage`.

\*/

- (void)finishSendingMessageAnimated:(BOOL)animated;

/\*\*

\* Animates the receiving of a new message. See `finishReceivingMessageAnimated:` for more details.

\*

\* @see `finishReceivingMessageAnimated:`.

\*/

- (void)finishReceivingMessage;

/\*\*

\* Completes the "receiving" of a new message by showing the typing indicator, adding a new collection view cell in the collection view,

\* reloading the collection view, and scrolling to the newly sent message as specified by `automaticallyScrollsToMostRecentMessage`.

\* Scrolling to the new message can be animated as specified by the animated parameter.

\*

\* @param animated Specifies whether the receiving of a message should be animated or not. Pass `YES` to animate changes, `NO` otherwise.

\*

\* @discussion You should call this method after adding a new "received" message to your data source and performing any related tasks.

\*

\* @see `automaticallyScrollsToMostRecentMessage`.

\*/

- (void)finishReceivingMessageAnimated:(BOOL)animated;

/\*\*

\* Scrolls the collection view such that the bottom most cell is completely visible, above the `inputToolbar`.

\*

\* @param animated Pass `YES` if you want to animate scrolling, `NO` if it should be immediate.

\*/

- (void)scrollToBottomAnimated:(BOOL)animated;

/\*\*

\* Used to decide if a message is incoming or outgoing.

\*

\* @discussion The default implementation of this method compares the `senderId` of the message to the

\* value of the `senderId` property and returns `YES` if they are equal. Subclasses can override

\* this method to specialize the decision logic.

\*/

- (BOOL)isOutgoingMessage:(id<JSQMessageData>)messageItem;

/\*\*

\* Scrolls the collection view so that the cell at the specified indexPath is completely visible above the `inputToolbar`.

\*

\* @param indexPath The indexPath for the cell that will be visible.

\* @param animated Pass `YES` if you want to animate scrolling, `NO` otherwise.

\*/

- (void)scrollToIndexPath:(NSIndexPath \*)indexPath animated:(BOOL)animated;

/\*\*

Call to super required.

\*/

- (void)viewDidLoad NS\_REQUIRES\_SUPER;

/\*\*

Call to super required.

\*/

- (void)viewWillAppear:(BOOL)animated NS\_REQUIRES\_SUPER;

/\*\*

Call to super required.

\*/

- (void)viewDidAppear:(BOOL)animated NS\_REQUIRES\_SUPER;

/\*\*

Call to super required.

\*/

- (void)viewWillDisappear:(BOOL)animated NS\_REQUIRES\_SUPER;

/\*\*

Call to super required.

\*/

- (void)viewDidDisappear:(BOOL)animated NS\_REQUIRES\_SUPER;

/\*\*

Called when `UIMenuControllerWillShowMenuNotification` is posted.

@param notification The posted notification.

\*/

- (void)didReceiveMenuWillShowNotification:(NSNotification \*)notification;

/\*\*

Called when `UIMenuControllerWillHideMenuNotification` is posted.

@param notification The posted notification.

\*/

- (void)didReceiveMenuWillHideNotification:(NSNotification \*)notification;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

#import "JSQMessagesAvatarImage.h"

/\*\*

\* `JSQMessagesAvatarImageFactory` is a factory that provides a means for creating and styling

\* `JSQMessagesAvatarImage` objects to be displayed in a `JSQMessagesCollectionViewCell` of a `JSQMessagesCollectionView`.

\*/

@interface JSQMessagesAvatarImageFactory : NSObject

/\*\*

\* Creates and returns a `JSQMessagesAvatarImage` object with the specified placeholderImage that is

\* cropped to a circle of the given diameter.

\*

\* @param placeholderImage An image object that represents a placeholder avatar image. This value must not be `nil`.

\* @param diameter An integer value specifying the diameter size of the avatar in points. This value must be greater than `0`.

\*

\* @return An initialized `JSQMessagesAvatarImage` object if created successfully, `nil` otherwise.

\*/

+ (JSQMessagesAvatarImage \*)avatarImageWithPlaceholder:(UIImage \*)placeholderImage diameter:(NSUInteger)diameter;

/\*\*

\* Creates and returns a `JSQMessagesAvatarImage` object with the specified image that is

\* cropped to a circle of the given diameter and used for the `avatarImage` and `avatarPlaceholderImage` properties

\* of the returned `JSQMessagesAvatarImage` object. This image is then copied and has a transparent black mask applied to it,

\* which is used for the `avatarHighlightedImage` property of the returned `JSQMessagesAvatarImage` object.

\*

\* @param image An image object that represents an avatar image. This value must not be `nil`.

\* @param diameter An integer value specifying the diameter size of the avatar in points. This value must be greater than `0`.

\*

\* @return An initialized `JSQMessagesAvatarImage` object if created successfully, `nil` otherwise.

\*/

+ (JSQMessagesAvatarImage \*)avatarImageWithImage:(UIImage \*)image diameter:(NSUInteger)diameter;

/\*\*

\* Returns a copy of the specified image that is cropped to a circle with the given diameter.

\*

\* @param image The image to crop. This value must not be `nil`.

\* @param diameter An integer value specifying the diameter size of the image in points. This value must be greater than `0`.

\*

\* @return A new image object if successful, `nil` otherwise.

\*/

+ (UIImage \*)circularAvatarImage:(UIImage \*)image withDiameter:(NSUInteger)diameter;

/\*\*

\* Returns a copy of the specified image that is cropped to a circle with the given diameter.

\* Additionally, a transparent overlay is applied to the image to represent a pressed or highlighted state.

\*

\* @param image The image to crop. This value must not be `nil`.

\* @param diameter An integer value specifying the diameter size of the image in points. This value must be greater than `0`.

\*

\* @return A new image object if successful, `nil` otherwise.

\*/

+ (UIImage \*)circularAvatarHighlightedImage:(UIImage \*)image withDiameter:(NSUInteger)diameter;

/\*\*

\* Creates and returns a `JSQMessagesAvatarImage` object with a circular shape that displays the specified userInitials

\* with the given backgroundColor, textColor, font, and diameter.

\*

\* @param userInitials The user initials to display in the avatar image. This value must not be `nil`.

\* @param backgroundColor The background color of the avatar. This value must not be `nil`.

\* @param textColor The color of the text of the userInitials. This value must not be `nil`.

\* @param font The font applied to userInitials. This value must not be `nil`.

\* @param diameter The diameter of the avatar image. This value must be greater than `0`.

\*

\* @return An initialized `JSQMessagesAvatarImage` object if created successfully, `nil` otherwise.

\*

\* @discussion This method does not attempt to detect or correct incompatible parameters.

\* That is to say, you are responsible for providing a font size and diameter that make sense.

\* For example, a font size of `14.0f` and a diameter of `34.0f` will result in an avatar similar to Messages in iOS 7.

\* However, a font size `30.0f` and diameter of `10.0f` will not produce a desirable image.

\* Further, this method does not check the length of userInitials. It is recommended that you pass a string of length `2` or `3`.

\*/

+ (JSQMessagesAvatarImage \*)avatarImageWithUserInitials:(NSString \*)userInitials

backgroundColor:(UIColor \*)backgroundColor

textColor:(UIColor \*)textColor

font:(UIFont \*)font

diameter:(NSUInteger)diameter;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

#import "JSQMessagesBubbleImage.h"

/\*\*

\* `JSQMessagesBubbleImageFactory` is a factory that provides a means for creating and styling

\* `JSQMessagesBubbleImage` objects to be displayed in a `JSQMessagesCollectionViewCell` of a `JSQMessagesCollectionView`.

\*/

@interface JSQMessagesBubbleImageFactory : NSObject

/\*\*

\* Creates and returns a new instance of `JSQMessagesBubbleImageFactory` that uses the

\* default bubble image assets and cap insets.

\*

\* @return An initialized `JSQMessagesBubbleImageFactory` object if created successfully, `nil` otherwise.

\*/

- (instancetype)init;

/\*\*

\* Creates and returns a new instance of `JSQMessagesBubbleImageFactory` having the specified

\* bubbleImage and capInsets. These values are used internally in the factory to produce

\* `JSQMessagesBubbleImage` objects.

\*

\* @param bubbleImage A template bubble image from which all images will be generated.

\* The image should represent the \*outgoing\* message bubble image, which will be flipped

\* horizontally for generating the corresponding \*incoming\* message bubble images. This value must not be `nil`.

\*

\* @param capInsets The values to use for the cap insets that define the unstretchable regions of the image.

\* Specify `UIEdgeInsetsZero` to have the factory create insets that allow the image to stretch from its center point.

\*

\* @return An initialized `JSQMessagesBubbleImageFactory` object if created successfully, `nil` otherwise.

\*/

- (instancetype)initWithBubbleImage:(UIImage \*)bubbleImage capInsets:(UIEdgeInsets)capInsets;

/\*\*

\* Creates and returns a `JSQMessagesBubbleImage` object with the specified color for \*outgoing\* message image bubbles.

\* The `messageBubbleImage` property of the `JSQMessagesBubbleImage` is configured with a flat bubble image, masked to the given color.

\* The `messageBubbleHighlightedImage` property is configured similarly, but with a darkened version of the given color.

\*

\* @param color The color of the bubble image in the image view. This value must not be `nil`.

\*

\* @return An initialized `JSQMessagesBubbleImage` object if created successfully, `nil` otherwise.

\*/

- (JSQMessagesBubbleImage \*)outgoingMessagesBubbleImageWithColor:(UIColor \*)color;

/\*\*

\* Creates and returns a `JSQMessagesBubbleImage` object with the specified color for \*incoming\* message image bubbles.

\* The `messageBubbleImage` property of the `JSQMessagesBubbleImage` is configured with a flat bubble image, masked to the given color.

\* The `messageBubbleHighlightedImage` property is configured similarly, but with a darkened version of the given color.

\*

\* @param color The color of the bubble image in the image view. This value must not be `nil`.

\*

\* @return An initialized `JSQMessagesBubbleImage` object if created successfully, `nil` otherwise.

\*/

- (JSQMessagesBubbleImage \*)incomingMessagesBubbleImageWithColor:(UIColor \*)color;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

@class JSQMessagesBubbleImageFactory;

/\*\*

\* An instance of `JSQMessagesMediaViewBubbleImageMasker` is an object that masks

\* media views for a `JSQMessageMediaData` object. Given a view, it will mask the view

\* with a bubble image for an outgoing or incoming media view.

\*

\* @see JSQMessageMediaData.

\* @see JSQMessagesBubbleImageFactory.

\* @see JSQMessagesBubbleImage.

\*/

@interface JSQMessagesMediaViewBubbleImageMasker : NSObject

/\*\*

\* Returns the bubble image factory that the masker uses to mask media views.

\*/

@property (strong, nonatomic, readonly) JSQMessagesBubbleImageFactory \*bubbleImageFactory;

/\*\*

\* Creates and returns a new instance of `JSQMessagesMediaViewBubbleImageMasker`

\* that uses a default instance of `JSQMessagesBubbleImageFactory`. The masker uses the `JSQMessagesBubbleImage`

\* objects returned by the factory to mask media views.

\*

\* @return An initialized `JSQMessagesMediaViewBubbleImageMasker` object if created successfully, `nil` otherwise.

\*

\* @see JSQMessagesBubbleImageFactory.

\* @see JSQMessagesBubbleImage.

\*/

- (instancetype)init;

/\*\*

\* Creates and returns a new instance of `JSQMessagesMediaViewBubbleImageMasker`

\* having the specified bubbleImageFactory. The masker uses the `JSQMessagesBubbleImage`

\* objects returned by the factory to mask media views.

\*

\* @param bubbleImageFactory An initialized `JSQMessagesBubbleImageFactory` object to use for masking media views. This value must not be `nil`.

\*

\* @return An initialized `JSQMessagesMediaViewBubbleImageMasker` object if created successfully, `nil` otherwise.

\*

\* @see JSQMessagesBubbleImageFactory.

\* @see JSQMessagesBubbleImage.

\*/

- (instancetype)initWithBubbleImageFactory:(JSQMessagesBubbleImageFactory \*)bubbleImageFactory NS\_DESIGNATED\_INITIALIZER;

/\*\*

\* Applies an outgoing bubble image mask to the specified mediaView.

\*

\* @param mediaView The media view to mask.

\*/

- (void)applyOutgoingBubbleImageMaskToMediaView:(UIView \*)mediaView;

/\*\*

\* Applies an incoming bubble image mask to the specified mediaView.

\*

\* @param mediaView The media view to mask.

\*/

- (void)applyIncomingBubbleImageMaskToMediaView:(UIView \*)mediaView;

/\*\*

\* A convenience method for applying a bubble image mask to the specified mediaView.

\* This method uses the default instance of `JSQMessagesBubbleImageFactory`.

\*

\* @param mediaView The media view to mask.

\* @param isOutgoing A boolean value specifiying whether or not the mask should be for an outgoing or incoming view.

\* Specify `YES` for outgoing and `NO` for incoming.

\*/

+ (void)applyBubbleImageMaskToMediaView:(UIView \*)mediaView isOutgoing:(BOOL)isOutgoing;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

/\*\*

\* An instance of `JSQMessagesTimestampFormatter` is a singleton object that provides an efficient means

\* for creating attributed and non-attributed string representations of `NSDate` objects.

\* It is intended to be used as the method by which you display timestamps in a `JSQMessagesCollectionView`.

\*/

@interface JSQMessagesTimestampFormatter : NSObject

/\*\*

\* Returns the cached date formatter object used by the `JSQMessagesTimestampFormatter` shared instance.

\*/

@property (strong, nonatomic, readonly) NSDateFormatter \*dateFormatter;

/\*\*

\* The text attributes to apply to the day, month, and year components of the string representation of a given date.

\* The default value is a dictionary containing attributes that specify centered, light gray text and the bold system font at size `12.0f`.

\*/

@property (copy, nonatomic) NSDictionary \*dateTextAttributes;

/\*\*

\* The text attributes to apply to the minute and hour componenents of the string representation of a given date.

\* The default value is a dictionary containing attributes that specify centered, light gray text and the system font at size `12.0f`.

\*/

@property (copy, nonatomic) NSDictionary \*timeTextAttributes;

/\*\*

\* Returns the shared timestamp formatter object.

\*

\* @return The shared timestamp formatter object.

\*/

+ (JSQMessagesTimestampFormatter \*)sharedFormatter;

/\*\*

\* Returns a string representation of the given date formatted in the current locale using `NSDateFormatterMediumStyle` for the date style

\* and `NSDateFormatterShortStyle` for the time style. It uses relative date formatting where possible.

\*

\* @param date The date to format.

\*

\* @return A formatted string representation of date.

\*/

- (NSString \*)timestampForDate:(NSDate \*)date;

/\*\*

\* Returns an attributed string representation of the given date formatted as described in `timestampForDate:`.

\* It applies the attributes in `dateTextAttributes` and `timeTextAttributes`, respectively.

\*

\* @param date The date to format.

\*

\* @return A formatted, attributed string representation of date.

\*

\* @see `timestampForDate:`.

\* @see `dateTextAttributes`.

\* @see `timeTextAttributes`.

\*/

- (NSAttributedString \*)attributedTimestampForDate:(NSDate \*)date;

/\*\*

\* Returns a string representation of \*only\* the minute and hour components of the given date formatted in the current locale styled using `NSDateFormatterShortStyle`.

\*

\* @param date The date to format.

\*

\* @return A formatted string representation of the minute and hour components of date.

\*/

- (NSString \*)timeForDate:(NSDate \*)date;

/\*\*

\* Returns a string representation of \*only\* the day, month, and year components of the given date formatted in the current locale styled using `NSDateFormatterMediumStyle`.

\*

\* @param date The date to format.

\*

\* @return A formatted string representation of the day, month, and year components of date.

\*/

- (NSString \*)relativeDateForDate:(NSDate \*)date;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

/\*\*

\* `JSQMessagesToolbarButtonFactory` is a factory that provides a means for creating the default

\* toolbar button items to be displayed in the content view of a `JSQMessagesInputToolbar`.

\*/

@interface JSQMessagesToolbarButtonFactory : NSObject

/\*\*

\* Creates and returns a new button that is styled as the default accessory button.

\* The button has a paper clip icon image and no text.

\*

\* @return A newly created button.

\*/

+ (UIButton \*)defaultAccessoryButtonItem;

/\*\*

\* Creates and returns a new button that is styled as the default send button.

\* The button has title text `@"Send"` and no image.

\*

\* @return A newly created button.

\*/

+ (UIButton \*)defaultSendButtonItem;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#ifndef JSQMessages\_JSQMessages\_h

#define JSQMessages\_JSQMessages\_h

#import "JSQMessagesViewController.h"

// Views

#import "JSQMessagesCollectionView.h"

#import "JSQMessagesCollectionViewCellIncoming.h"

#import "JSQMessagesCollectionViewCellOutgoing.h"

#import "JSQMessagesTypingIndicatorFooterView.h"

#import "JSQMessagesLoadEarlierHeaderView.h"

// Layout

#import "JSQMessagesBubbleSizeCalculating.h"

#import "JSQMessagesBubblesSizeCalculator.h"

#import "JSQMessagesCollectionViewFlowLayout.h"

#import "JSQMessagesCollectionViewLayoutAttributes.h"

#import "JSQMessagesCollectionViewFlowLayoutInvalidationContext.h"

#import "JSQAudioMediaViewAttributes.h"

// Toolbar

#import "JSQMessagesComposerTextView.h"

#import "JSQMessagesInputToolbar.h"

#import "JSQMessagesToolbarContentView.h"

// Model

#import "JSQMessage.h"

#import "JSQMediaItem.h"

#import "JSQAudioMediaItem.h"

#import "JSQPhotoMediaItem.h"

#import "JSQLocationMediaItem.h"

#import "JSQVideoMediaItem.h"

#import "JSQMessagesBubbleImage.h"

#import "JSQMessagesAvatarImage.h"

#import "JSQAudioMediaViewAttributes.h"

// Protocols

#import "JSQMessageData.h"

#import "JSQMessageMediaData.h"

#import "JSQMessageAvatarImageDataSource.h"

#import "JSQMessageBubbleImageDataSource.h"

#import "JSQMessagesCollectionViewDataSource.h"

#import "JSQMessagesCollectionViewDelegateFlowLayout.h"

// Factories

#import "JSQMessagesAvatarImageFactory.h"

#import "JSQMessagesBubbleImageFactory.h"

#import "JSQMessagesMediaViewBubbleImageMasker.h"

#import "JSQMessagesTimestampFormatter.h"

#import "JSQMessagesToolbarButtonFactory.h"

// Categories

#import "JSQSystemSoundPlayer+JSQMessages.h"

#import "NSString+JSQMessages.h"

#import "UIColor+JSQMessages.h"

#import "UIImage+JSQMessages.h"

#import "UIView+JSQMessages.h"

#import "NSBundle+JSQMessages.h"

#endif

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

NS\_ASSUME\_NONNULL\_BEGIN

@class JSQMessagesCollectionView;

/\*\*

\* The `JSQMessagesViewAccessoryButtonDelegate` protocol defines methods that allow you to

\* handle accessory actions for the collection view.

\*/

@protocol JSQMessagesViewAccessoryButtonDelegate <NSObject>

@required

/\*\*

\* Notifies the delegate that the accessory button at the specified indexPath did receive a tap event.

\*

\* @param messageView The collection view object that is notifying the delegate of the tap event.

\* @param path The index path of the item for which the accessory button was tapped.

\*/

- (void)messageView:(JSQMessagesCollectionView \*)messageView didTapAccessoryButtonAtIndexPath:(NSIndexPath \*)path;

@end

NS\_ASSUME\_NONNULL\_END

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <AVFoundation/AVFoundation.h>

#import <UIKit/UIKit.h>

NS\_ASSUME\_NONNULL\_BEGIN

/\*\*

An instance of `JSQAudioMediaViewAttributes` specifies the appearance configuration of a `JSQAudioMediaItem`.

Use this class to customize the appearance of `JSQAudioMediaItem`.

\*/

@interface JSQAudioMediaViewAttributes : NSObject

/\*\*

\* The image for the play button. The default is a play icon.

\*/

@property (nonatomic, strong) UIImage \*playButtonImage;

/\*\*

\* The image for the pause button. The default is a pause icon.

\*/

@property (nonatomic, strong) UIImage \*pauseButtonImage;

/\*\*

\* The font for the elapsed time label. The default is a system font.

\*/

@property (strong, nonatomic) UIFont \*labelFont;

/\*\*

\* Specifies whether to show fractions of a second for audio files with a duration of less than 1 minute.

\*/

@property (nonatomic, assign) BOOL showFractionalSeconds;

/\*\*

\* The background color for the player.

\*/

@property (nonatomic, strong) UIColor \*backgroundColor;

/\*\*

\* The tint color for the player.

\*/

@property (nonatomic, strong) UIColor \*tintColor;

/\*\*

\* Insets that sepcify the padding around the play/pause button and time label.

\*/

@property (nonatomic, assign) UIEdgeInsets controlInsets;

/\*\*

\* Specifies the padding between the button, progress bar, and label.

\*/

@property (nonatomic, assign) CGFloat controlPadding;

/\*\*

\* Specifies the audio category set prior to playback.

\*/

@property (nonatomic, copy) NSString \*audioCategory;

/\*\*

\* Specifies the audio category options set prior to playback.

\*/

@property (nonatomic) AVAudioSessionCategoryOptions audioCategoryOptions;

/\*\*

Initializes and returns a `JSQAudioMediaViewAttributes` instance having the specified attributes.

@param playButtonImage The image for the play button.

@param pauseButtonImage The image for the pause button.

@param labelFont The font for the elapsed time label.

@param showFractionalSeconds Specifies whether to show fractions of a second for audio files with a duration of less than 1 minute.

@param backgroundColor The background color for the player.

@param tintColor The tint color for the player.

@param controlInsets Insets that sepcify the padding around the play/pause button and time label.

@param controlPadding Specifies the padding between the button, progress bar, and label.

@param audioCategory Specifies the audio category set prior to playback.

@param audioCategoryOptions Specifies the audio category options set prior to playback.

@return A new `JSQAudioMediaViewAttributes` instance

\*/

- (instancetype)initWithPlayButtonImage:(UIImage \*)playButtonImage

pauseButtonImage:(UIImage \*)pauseButtonImage

labelFont:(UIFont \*)labelFont

showFractionalSecodns:(BOOL)showFractionalSeconds

backgroundColor:(UIColor \*)backgroundColor

tintColor:(UIColor \*)tintColor

controlInsets:(UIEdgeInsets)controlInsets

controlPadding:(CGFloat)controlPadding

audioCategory:(NSString \*)audioCategory

audioCategoryOptions:(AVAudioSessionCategoryOptions)audioCategoryOptions NS\_DESIGNATED\_INITIALIZER;

/\*\*

Initializes and returns a default `JSQAudioMediaViewAttributes` instance.

@return A new `JSQAudioMediaViewAttributes` instance

\*/

- (instancetype)init;

@end

NS\_ASSUME\_NONNULL\_END

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

@class JSQMessagesCollectionViewFlowLayout;

@protocol JSQMessageData;

/\*\*

\* The `JSQMessagesBubbleSizeCalculating` protocol defines the common interface through which

\* an object provides layout information to an instance of `JSQMessagesCollectionViewFlowLayout`.

\*

\* A concrete class that conforms to this protocol is provided in the library.

\* See `JSQMessagesBubbleSizeCalculator`.

\*/

@protocol JSQMessagesBubbleSizeCalculating <NSObject>

/\*\*

\* Computes and returns the size of the `messageBubbleImageView` property

\* of a `JSQMessagesCollectionViewCell` for the specified messageData at indexPath.

\*

\* @param messageData A message data object.

\* @param indexPath The index path at which messageData is located.

\* @param layout The layout object asking for this information.

\*

\* @return A sizes that specifies the required dimensions to display the entire message contents.

\* Note, this is \*not\* the entire cell, but only its message bubble.

\*/

- (CGSize)messageBubbleSizeForMessageData:(id<JSQMessageData>)messageData

atIndexPath:(NSIndexPath \*)indexPath

withLayout:(JSQMessagesCollectionViewFlowLayout \*)layout;

/\*\*

\* Notifies the receiver that the layout will be reset.

\* Use this method to clear any cached layout information, if necessary.

\*

\* @param layout The layout object notifying the receiver.

\*/

- (void)prepareForResettingLayout:(JSQMessagesCollectionViewFlowLayout \*)layout;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import "JSQMessagesBubbleSizeCalculating.h"

/\*\*

\* An instance of `JSQMessagesBubblesSizeCalculator` is responsible for calculating

\* message bubble sizes for an instance of `JSQMessagesCollectionViewFlowLayout`.

\*/

@interface JSQMessagesBubblesSizeCalculator : NSObject <JSQMessagesBubbleSizeCalculating>

/\*\*

\* Initializes and returns a bubble size calculator with the given cache and minimumBubbleWidth.

\*

\* @param cache A cache object used to store layout information.

\* @param minimumBubbleWidth The minimum width for any given message bubble.

\* @param usesFixedWidthBubbles Specifies whether or not to use fixed-width bubbles.

\* If `NO` (the default), then bubbles will resize when rotating to landscape.

\*

\* @return An initialized `JSQMessagesBubblesSizeCalculator` object if successful, `nil` otherwise.

\*/

- (instancetype)initWithCache:(NSCache \*)cache

minimumBubbleWidth:(NSUInteger)minimumBubbleWidth

usesFixedWidthBubbles:(BOOL)usesFixedWidthBubbles NS\_DESIGNATED\_INITIALIZER;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

//

// Ideas for springy collection view layout taken from Ash Furrow

// ASHSpringyCollectionView

// https://github.com/AshFurrow/ASHSpringyCollectionView

//

#import <UIKit/UIKit.h>

#import "JSQMessagesBubbleSizeCalculating.h"

@class JSQMessagesCollectionView;

/\*\*

\* A constant that describes the default height for all label subviews in a `JSQMessagesCollectionViewCell`.

\*

\* @see JSQMessagesCollectionViewCell.

\*/

FOUNDATION\_EXPORT const CGFloat kJSQMessagesCollectionViewCellLabelHeightDefault;

/\*\*

\* A constant that describes the default size for avatar images in a `JSQMessagesCollectionViewFlowLayout`.

\*/

FOUNDATION\_EXPORT const CGFloat kJSQMessagesCollectionViewAvatarSizeDefault;

/\*\*

\* The `JSQMessagesCollectionViewFlowLayout` is a concrete layout object that inherits

\* from `UICollectionViewFlowLayout` and organizes message items in a vertical list.

\* Each `JSQMessagesCollectionViewCell` in the layout can display messages of arbitrary sizes and avatar images,

\* as well as metadata such as a timestamp and sender.

\* You can easily customize the layout via its properties or its delegate methods

\* defined in `JSQMessagesCollectionViewDelegateFlowLayout`.

\*

\* @see JSQMessagesCollectionViewDelegateFlowLayout.

\* @see JSQMessagesCollectionViewCell.

\*/

@interface JSQMessagesCollectionViewFlowLayout : UICollectionViewFlowLayout

/\*\*

\* The collection view object currently using this layout object.

\*/

// TODO: fix, rename "messagesCollectionView", see #920

#pragma clang diagnostic push

#pragma clang diagnostic ignored "-Wincompatible-property-type"

@property (readonly, nonatomic) JSQMessagesCollectionView \*collectionView;

#pragma clang diagnostic pop

/\*\*

\* The object that the layout uses to calculate bubble sizes.

\* The default value is an instance of `JSQMessagesBubblesSizeCalculator`.

\*/

@property (strong, nonatomic) id<JSQMessagesBubbleSizeCalculating> bubbleSizeCalculator;

/\*\*

\* Specifies whether or not the layout should enable spring behavior dynamics for its items using `UIDynamics`.

\*

\* @discussion The default value is `NO`, which disables "springy" or "bouncy" items in the layout.

\* Set to `YES` if you want items to have spring behavior dynamics. You \*must\* set this property from `viewDidAppear:`

\* in your `JSQMessagesViewController` subclass.

\*

\* @warning Though this feature is mostly stable, it is still considered an experimental feature.

\*/

@property (assign, nonatomic) BOOL springinessEnabled;

/\*\*

\* Specifies the degree of resistence for the "springiness" of items in the layout.

\* This property has no effect if `springinessEnabled` is set to `NO`.

\*

\* @discussion The default value is `1000`. Increasing this value increases the resistance, that is, items become less "bouncy".

\* Decrease this value in order to make items more "bouncy".

\*/

@property (assign, nonatomic) NSUInteger springResistanceFactor;

/\*\*

\* Returns the width of items in the layout.

\*/

@property (readonly, nonatomic) CGFloat itemWidth;

/\*\*

\* The font used to display the body a text message in the message bubble of each

\* `JSQMessagesCollectionViewCell` in the collectionView.

\*

\* @discussion The default value is the preferred system font for `UIFontTextStyleBody`. This value must not be `nil`.

\*/

@property (strong, nonatomic) UIFont \*messageBubbleFont;

/\*\*

\* The horizontal spacing used to lay out the `messageBubbleContainerView` frame within each `JSQMessagesCollectionViewCell`.

\* This container view holds the message bubble image and message contents of a cell.

\*

\* This value specifies the horizontal spacing between the `messageBubbleContainerView` and

\* the edge of the collection view cell in which it is displayed. That is, the edge that is opposite the avatar image.

\*

\* @discussion The default value is `40.0f` on iPhone and `240.0f` on iPad. This value must be positive.

\* For \*outgoing\* messages, this value specifies the amount of spacing from the left most

\* edge of the collectionView to the left most edge of a message bubble within a cell.

\*

\* For \*incoming\* messages, this value specifies the amount of spacing from the right most

\* edge of the collectionView to the right most edge of a message bubble within a cell.

\*

\* @warning This value may not be exact when the layout object finishes laying out its items, due to the constraints it must satisfy.

\* This value should be considered more of a recommendation or suggestion to the layout, not an exact value.

\*

\* @see JSQMessagesCollectionViewCellIncoming.

\* @see JSQMessagesCollectionViewCellOutgoing.

\*/

@property (assign, nonatomic) CGFloat messageBubbleLeftRightMargin;

/\*\*

\* The inset of the frame of the text view within the `messageBubbleContainerView` of each `JSQMessagesCollectionViewCell`.

\* The inset values should be positive and are applied in the following ways:

\*

\* 1. The right value insets the text view frame on the side adjacent to the avatar image

\* (or where the avatar would normally appear). For outgoing messages this is the right side,

\* for incoming messages this is the left side.

\*

\* 2. The left value insets the text view frame on the side opposite the avatar image

\* (or where the avatar would normally appear). For outgoing messages this is the left side,

\* for incoming messages this is the right side.

\*

\* 3. The top value insets the top of the frame.

\*

\* 4. The bottom value insets the bottom of the frame.

\*

\* @discussion The default value is `{0.0f, 0.0f, 0.0f, 6.0f}`.

\*

\* @warning Adjusting this value is an advanced endeavour and not recommended.

\* You will only need to adjust this value should you choose to provide your own bubble image assets.

\* Changing this value may also require you to manually calculate the itemSize for each cell

\* in the layout by overriding the delegate method `collectionView:layout:sizeForItemAtIndexPath:`

\*/

@property (assign, nonatomic) UIEdgeInsets messageBubbleTextViewFrameInsets;

/\*\*

\* The inset of the text container's layout area within the text view's content area in each `JSQMessagesCollectionViewCell`.

\* The specified inset values should be positive.

\*

\* @discussion The default value is `{7.0f, 14.0f, 7.0f, 14.0f}`.

\*

\* @warning Adjusting this value is an advanced endeavour and not recommended.

\* You will only need to adjust this value should you choose to provide your own bubble image assets.

\* Changing this value may also require you to manually calculate the itemSize for each cell

\* in the layout by overriding the delegate method `collectionView:layout:sizeForItemAtIndexPath:`

\*/

@property (assign, nonatomic) UIEdgeInsets messageBubbleTextViewTextContainerInsets;

/\*\*

\* The size of the avatar image view for incoming messages.

\*

\* @discussion The default value is `(30.0f, 30.0f)`. Set to `CGSizeZero` to remove incoming avatars.

\* You may use `kJSQMessagesCollectionViewAvatarSizeDefault` to size your avatars to the default value.

\*/

@property (assign, nonatomic) CGSize incomingAvatarViewSize;

/\*\*

\* The size of the avatar image view for outgoing messages.

\*

\* @discussion The default value is `(30.0f, 30.0f)`. Set to `CGSizeZero` to remove outgoing avatars.

\* You may use `kJSQMessagesCollectionViewAvatarSizeDefault` to size your avatars to the default value.

\*/

@property (assign, nonatomic) CGSize outgoingAvatarViewSize;

/\*\*

\* The maximum number of items that the layout should keep in its cache of layout information.

\*

\* @discussion The default value is `200`. A limit of `0` means no limit. This is not a strict limit.

\*/

@property (assign, nonatomic) NSUInteger cacheLimit;

/\*\*

\* Computes and returns the size of the `messageBubbleImageView` property of a `JSQMessagesCollectionViewCell`

\* at the specified indexPath.

\*

\* @param indexPath The index path of the item to be displayed.

\*

\* @return The size of the message bubble for the item displayed at indexPath.

\*

\* @discussion The layout uses its `bubbleSizeCalculator` object to perform this computation.

\* The returned size contains the required dimensions to display the entire message contents.

\* Note, this is \*not\* the entire cell, but only its message bubble.

\*/

- (CGSize)messageBubbleSizeForItemAtIndexPath:(NSIndexPath \*)indexPath;

/\*\*

\* Computes and returns the size of the item specified by indexPath.

\*

\* @param indexPath The index path of the item to be displayed.

\*

\* @return The size of the item displayed at indexPath.

\*/

- (CGSize)sizeForItemAtIndexPath:(NSIndexPath \*)indexPath;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <UIKit/UIKit.h>

/\*\*

\* A `JSQMessagesCollectionViewFlowLayoutInvalidationContext` object specifies properties for

\* determining whether to recompute the size of items or their position in the layout.

\* The flow layout object creates instances of this class when it needs to invalidate its contents

\* in response to changes. You can also create instances when invalidating the flow layout manually.

\*

\*/

@interface JSQMessagesCollectionViewFlowLayoutInvalidationContext : UICollectionViewFlowLayoutInvalidationContext

/\*\*

\* A boolean indicating whether to empty the messages layout information cache for items and views in the layout.

\* The default value is `NO`.

\*/

@property (nonatomic, assign) BOOL invalidateFlowLayoutMessagesCache;

/\*\*

\* Creates and returns a new `JSQMessagesCollectionViewFlowLayoutInvalidationContext` object.

\*

\* @discussion When you need to invalidate the `JSQMessagesCollectionViewFlowLayout` object for your

\* `JSQMessagesViewController` subclass, you should use this method to instantiate a new invalidation

\* context and pass this object to `invalidateLayoutWithContext:`.

\*

\* @return An initialized invalidation context object if successful, otherwise `nil`.

\*/

+ (instancetype)context;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <UIKit/UIKit.h>

/\*\*

\* A `JSQMessagesCollectionViewLayoutAttributes` is an object that manages the layout-related attributes

\* for a given `JSQMessagesCollectionViewCell` in a `JSQMessagesCollectionView`.

\*/

@interface JSQMessagesCollectionViewLayoutAttributes : UICollectionViewLayoutAttributes <NSCopying>

/\*\*

\* The font used to display the body of a text message in a message bubble within a `JSQMessagesCollectionViewCell`.

\* This value must not be `nil`.

\*/

@property (strong, nonatomic) UIFont \*messageBubbleFont;

/\*\*

\* The width of the `messageBubbleContainerView` of a `JSQMessagesCollectionViewCell`.

\* This value should be greater than `0.0`.

\*

\* @see JSQMessagesCollectionViewCell.

\*/

@property (assign, nonatomic) CGFloat messageBubbleContainerViewWidth;

/\*\*

\* The inset of the text container's layout area within the text view's content area in a `JSQMessagesCollectionViewCell`.

\* The specified inset values should be greater than or equal to `0.0`.

\*/

@property (assign, nonatomic) UIEdgeInsets textViewTextContainerInsets;

/\*\*

\* The inset of the frame of the text view within a `JSQMessagesCollectionViewCell`.

\*

\* @discussion The inset values should be greater than or equal to `0.0` and are applied in the following ways:

\*

\* 1. The right value insets the text view frame on the side adjacent to the avatar image

\* (or where the avatar would normally appear). For outgoing messages this is the right side,

\* for incoming messages this is the left side.

\*

\* 2. The left value insets the text view frame on the side opposite the avatar image

\* (or where the avatar would normally appear). For outgoing messages this is the left side,

\* for incoming messages this is the right side.

\*

\* 3. The top value insets the top of the frame.

\*

\* 4. The bottom value insets the bottom of the frame.

\*/

@property (assign, nonatomic) UIEdgeInsets textViewFrameInsets;

/\*\*

\* The size of the `avatarImageView` of a `JSQMessagesCollectionViewCellIncoming`.

\* The size values should be greater than or equal to `0.0`.

\*

\* @see JSQMessagesCollectionViewCellIncoming.

\*/

@property (assign, nonatomic) CGSize incomingAvatarViewSize;

/\*\*

\* The size of the `avatarImageView` of a `JSQMessagesCollectionViewCellOutgoing`.

\* The size values should be greater than or equal to `0.0`.

\*

\* @see `JSQMessagesCollectionViewCellOutgoing`.

\*/

@property (assign, nonatomic) CGSize outgoingAvatarViewSize;

/\*\*

\* The height of the `cellTopLabel` of a `JSQMessagesCollectionViewCell`.

\* This value should be greater than or equal to `0.0`.

\*

\* @see JSQMessagesCollectionViewCell.

\*/

@property (assign, nonatomic) CGFloat cellTopLabelHeight;

/\*\*

\* The height of the `messageBubbleTopLabel` of a `JSQMessagesCollectionViewCell`.

\* This value should be greater than or equal to `0.0`.

\*

\* @see JSQMessagesCollectionViewCell.

\*/

@property (assign, nonatomic) CGFloat messageBubbleTopLabelHeight;

/\*\*

\* The height of the `cellBottomLabel` of a `JSQMessagesCollectionViewCell`.

\* This value should be greater than or equal to `0.0`.

\*

\* @see JSQMessagesCollectionViewCell.

\*/

@property (assign, nonatomic) CGFloat cellBottomLabelHeight;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import "JSQMediaItem.h"

#import "JSQAudioMediaViewAttributes.h"

#import <AVFoundation/AVFoundation.h>

@class JSQAudioMediaItem;

NS\_ASSUME\_NONNULL\_BEGIN

@protocol JSQAudioMediaItemDelegate <NSObject>

/\*\*

\* Tells the delegate if the specified `JSQAudioMediaItem` changes the sound category or categoryOptions, or if an error occurs.

\*/

- (void)audioMediaItem:(JSQAudioMediaItem \*)audioMediaItem

didChangeAudioCategory:(NSString \*)category

options:(AVAudioSessionCategoryOptions)options

error:(nullable NSError \*)error;

@end

/\*\*

\* The `JSQAudioMediaItem` class is a concrete `JSQMediaItem` subclass that implements the `JSQMessageMediaData` protocol

\* and represents an audio media message. An initialized `JSQAudioMediaItem` object can be passed

\* to a `JSQMediaMessage` object during its initialization to construct a valid media message object.

\* You may wish to subclass `JSQAudioMediaItem` to provide additional functionality or behavior.

\*/

@interface JSQAudioMediaItem : JSQMediaItem <JSQMessageMediaData, AVAudioPlayerDelegate, NSCoding, NSCopying>

/\*\*

\* The delegate object for audio event notifications.

\*/

@property (nonatomic, weak, nullable) id<JSQAudioMediaItemDelegate> delegate;

/\*\*

\* The view attributes to configure the appearance of the audio media view.

\*/

@property (nonatomic, strong, readonly) JSQAudioMediaViewAttributes \*audioViewAttributes;

/\*\*

\* A data object that contains an audio resource.

\*/

@property (nonatomic, strong, nullable) NSData \*audioData;

/\*\*

\* Initializes and returns a audio media item having the given audioData.

\*

\* @param audioData The data object that contains the audio resource.

\* @param audioViewAttributes The view attributes to configure the appearance of the audio media view.

\*

\* @return An initialized `JSQAudioMediaItem`.

\*

\* @discussion If the audio must be downloaded from the network,

\* you may initialize a `JSQVideoMediaItem` with a `nil` audioData.

\* Once the audio is available you can set the `audioData` property.

\*/

- (instancetype)initWithData:(nullable NSData \*)audioData

audioViewAttributes:(JSQAudioMediaViewAttributes \*)audioViewAttributes NS\_DESIGNATED\_INITIALIZER;

/\*\*

\* Initializes and returns a default audio media item.

\*

\* @return An initialized `JSQAudioMediaItem`.

\*

\* @discussion You must set `audioData` to enable the play button.

\*/

- (instancetype)init;

/\*\*

Initializes and returns a default audio media using the specified view attributes.

@param audioViewAttributes The view attributes to configure the appearance of the audio media view.

@return An initialized `JSQAudioMediaItem`.

\*/

- (instancetype)initWithAudioViewAttributes:(JSQAudioMediaViewAttributes \*)audioViewAttributes;

/\*\*

\* Initializes and returns an audio media item having the given audioData.

\*

\* @param audioData The data object that contains the audio resource.

\*

\* @return An initialized `JSQAudioMediaItem`.

\*

\* @discussion If the audio must be downloaded from the network,

\* you may initialize a `JSQAudioMediaItem` with a `nil` audioData.

\* Once the audio is available you can set the `audioData` property.

\*/

- (instancetype)initWithData:(nullable NSData \*)audioData;

/\*\*

\* Sets or updates the data object in an audio media item with the data specified at audioURL.

\*

\* @param audioURL A File URL containing the location of the audio data.

\*/

- (void)setAudioDataWithUrl:(nonnull NSURL \*)audioURL;

@end

NS\_ASSUME\_NONNULL\_END

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <CoreLocation/CoreLocation.h>

#import <MapKit/MapKit.h>

/\*\*

\* A completion handler block for a `JSQLocationMediaItem`. See `setLocation: withCompletionHandler:`.

\*/

typedef void (^JSQLocationMediaItemCompletionBlock)(void);

#import "JSQMediaItem.h"

/\*\*

\* The `JSQLocationMediaItem` class is a concrete `JSQMediaItem` subclass that implements the `JSQMessageMediaData` protocol

\* and represents a location media message. An initialized `JSQLocationMediaItem` object can be passed

\* to a `JSQMediaMessage` object during its initialization to construct a valid media message object.

\* You may wish to subclass `JSQLocationMediaItem` to provide additional functionality or behavior.

\*/

@interface JSQLocationMediaItem : JSQMediaItem <JSQMessageMediaData, MKAnnotation, NSCoding, NSCopying>

/\*\*

\* The location for the media item. The default value is `nil`.

\*/

@property (copy, nonatomic) CLLocation \*location;

/\*\*

\* The coordinate of the location property.

\*/

@property (readonly, nonatomic) CLLocationCoordinate2D coordinate;

/\*\*

\* Initializes and returns a location media item object having the given location.

\*

\* @param location The location for the media item. This value may be `nil`.

\*

\* @return An initialized `JSQLocationMediaItem` if successful, `nil` otherwise.

\*

\* @discussion If the location data must be dowloaded from the network,

\* you may initialize a `JSQLocationMediaItem` object with a `nil` location.

\* Once the location data has been retrieved, you can then set the location property

\* using `setLocation: withCompletionHandler:`

\*/

- (instancetype)initWithLocation:(CLLocation \*)location;

/\*\*

\* Sets the specified location for the location media item and immediately begins creating

\* a map view snapshot image on a background thread. The map view zooms to a default region whose center point

\* is the location coordinate and whose span is 500 meters for both the latitudinal and longitudinal meters.

\*

\* The specified block is executed upon completion of creating the snapshot image and is executed on the app’s main thread.

\*

\* @param location The location for the media item.

\* @param completion The block to call after the map view snapshot for the given location has been created.

\*/

- (void)setLocation:(CLLocation \*)location withCompletionHandler:(JSQLocationMediaItemCompletionBlock)completion;

/\*\*

\* Sets the specified location for the location media item and immediately begins creating

\* a map view snapshot image on a background thread.

\*

\* The specified block is executed upon completion of creating the snapshot image and is executed on the app’s main thread.

\*

\* @param location The location for the media item.

\* @param region The map region that you want to capture.

\* @param completion The block to call after the map view snapshot for the given location has been created.

\*/

- (void)setLocation:(CLLocation \*)location

region:(MKCoordinateRegion)region withCompletionHandler:(JSQLocationMediaItemCompletionBlock)completion;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import "JSQMessageMediaData.h"

/\*\*

\* The `JSQMediaItem` class is an abstract base class for media item model objects that represents

\* a single media attachment for a user message. It provides some default behavior for media items,

\* including a default mediaViewDisplaySize, a default mediaPlaceholderView, and view masking as

\* specified by appliesMediaViewMaskAsOutgoing.

\*

\* @warning This class is intended to be subclassed. You should not use it directly.

\*

\* @see JSQLocationMediaItem.

\* @see JSQPhotoMediaItem.

\* @see JSQVideoMediaItem.

\*/

@interface JSQMediaItem : NSObject <JSQMessageMediaData, NSCoding, NSCopying>

/\*\*

\* A boolean value indicating whether this media item should apply

\* an outgoing or incoming bubble image mask to its media views.

\* Specify `YES` for an outgoing mask, and `NO` for an incoming mask.

\* The default value is `YES`.

\*/

@property (assign, nonatomic) BOOL appliesMediaViewMaskAsOutgoing;

/\*\*

\* Initializes and returns a media item with the specified value for maskAsOutgoing.

\*

\* @param maskAsOutgoing A boolean value indicating whether this media item should apply

\* an outgoing or incoming bubble image mask to its media views.

\*

\* @return An initialized `JSQMediaItem` object if successful, `nil` otherwise.

\*/

- (instancetype)initWithMaskAsOutgoing:(BOOL)maskAsOutgoing;

/\*\*

\* Clears any media view or media placeholder view that the item has cached.

\*/

- (void)clearCachedMediaViews;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import "JSQMessageData.h"

/\*\*

\* The `JSQMessage` class is a concrete class for message model objects that represents a single user message.

\* The message can be a text message or media message, depending on how it is initialized.

\* It implements the `JSQMessageData` protocol and it contains the senderId, senderDisplayName,

\* and the date that the message was sent. If initialized as a media message it also contains a media attachment,

\* otherwise it contains the message text.

\*/

@interface JSQMessage : NSObject <JSQMessageData, NSCoding, NSCopying>

/\*\*

\* Returns the string identifier that uniquely identifies the user who sent the message.

\*/

@property (copy, nonatomic, readonly) NSString \*senderId;

/\*\*

\* Returns the display name for the user who sent the message. This value does not have to be unique.

\*/

@property (copy, nonatomic, readonly) NSString \*senderDisplayName;

/\*\*

\* Returns the date that the message was sent.

\*/

@property (copy, nonatomic, readonly) NSDate \*date;

/\*\*

\* Returns a boolean value specifying whether or not the message contains media.

\* If `NO`, the message contains text. If `YES`, the message contains media.

\* The value of this property depends on how the object was initialized.

\*/

@property (assign, nonatomic, readonly) BOOL isMediaMessage;

/\*\*

\* Returns the body text of the message, or `nil` if the message is a media message.

\* That is, if `isMediaMessage` is equal to `YES` then this value will be `nil`.

\*/

@property (copy, nonatomic, readonly) NSString \*text;

/\*\*

\* Returns the media item attachment of the message, or `nil` if the message is not a media message.

\* That is, if `isMediaMessage` is equal to `NO` then this value will be `nil`.

\*/

@property (copy, nonatomic, readonly) id<JSQMessageMediaData> media;

#pragma mark - Initialization

/\*\*

\* Initializes and returns a message object having the given senderId, displayName, text,

\* and current system date.

\*

\* @param senderId The unique identifier for the user who sent the message. This value must not be `nil`.

\* @param displayName The display name for the user who sent the message. This value must not be `nil`.

\* @param text The body text of the message. This value must not be `nil`.

\*

\* @discussion Initializing a `JSQMessage` with this method will set `isMediaMessage` to `NO`.

\*

\* @return An initialized `JSQMessage` object if successful, `nil` otherwise.

\*/

+ (instancetype)messageWithSenderId:(NSString \*)senderId

displayName:(NSString \*)displayName

text:(NSString \*)text;

/\*\*

\* Initializes and returns a message object having the given senderId, senderDisplayName, date, and text.

\*

\* @param senderId The unique identifier for the user who sent the message. This value must not be `nil`.

\* @param senderDisplayName The display name for the user who sent the message. This value must not be `nil`.

\* @param date The date that the message was sent. This value must not be `nil`.

\* @param text The body text of the message. This value must not be `nil`.

\*

\* @discussion Initializing a `JSQMessage` with this method will set `isMediaMessage` to `NO`.

\*

\* @return An initialized `JSQMessage` object if successful, `nil` otherwise.

\*/

- (instancetype)initWithSenderId:(NSString \*)senderId

senderDisplayName:(NSString \*)senderDisplayName

date:(NSDate \*)date

text:(NSString \*)text;

/\*\*

\* Initializes and returns a message object having the given senderId, displayName, media,

\* and current system date.

\*

\* @param senderId The unique identifier for the user who sent the message. This value must not be `nil`.

\* @param displayName The display name for the user who sent the message. This value must not be `nil`.

\* @param media The media data for the message. This value must not be `nil`.

\*

\* @discussion Initializing a `JSQMessage` with this method will set `isMediaMessage` to `YES`.

\*

\* @return An initialized `JSQMessage` object if successful, `nil` otherwise.

\*/

+ (instancetype)messageWithSenderId:(NSString \*)senderId

displayName:(NSString \*)displayName

media:(id<JSQMessageMediaData>)media;

/\*\*

\* Initializes and returns a message object having the given senderId, displayName, date, and media.

\*

\* @param senderId The unique identifier for the user who sent the message. This value must not be `nil`.

\* @param senderDisplayName The display name for the user who sent the message. This value must not be `nil`.

\* @param date The date that the message was sent. This value must not be `nil`.

\* @param media The media data for the message. This value must not be `nil`.

\*

\* @discussion Initializing a `JSQMessage` with this method will set `isMediaMessage` to `YES`.

\*

\* @return An initialized `JSQMessage` object if successful, `nil` otherwise.

\*/

- (instancetype)initWithSenderId:(NSString \*)senderId

senderDisplayName:(NSString \*)senderDisplayName

date:(NSDate \*)date

media:(id<JSQMessageMediaData>)media;

/\*\*

\* Not a valid initializer.

\*/

- (id)init NS\_UNAVAILABLE;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

/\*\*

\* The `JSQMessageAvatarImageDataSource` protocol defines the common interface through which

\* a `JSQMessagesViewController` and `JSQMessagesCollectionView` interact with avatar image model objects.

\*

\* It declares the required and optional methods that a class must implement so that instances

\* of that class can be display properly within a `JSQMessagesCollectionViewCell`.

\*

\* A concrete class that conforms to this protocol is provided in the library. See `JSQMessagesAvatarImage`.

\*

\* @see JSQMessagesAvatarImage.

\*/

@protocol JSQMessageAvatarImageDataSource <NSObject>

@required

/\*\*

\* @return The avatar image for a regular display state.

\*

\* @discussion You may return `nil` from this method while the image is being downloaded.

\*/

- (UIImage \*)avatarImage;

/\*\*

\* @return The avatar image for a highlighted display state.

\*

\* @discussion You may return `nil` from this method if this does not apply.

\*/

- (UIImage \*)avatarHighlightedImage;

/\*\*

\* @return A placeholder avatar image to be displayed if avatarImage is not yet available, or `nil`.

\* For example, if avatarImage needs to be downloaded, this placeholder image

\* will be used until avatarImage is not `nil`.

\*

\* @discussion If you do not need support for a placeholder image, that is, your images

\* are stored locally on the device, then you may simply return the same value as avatarImage here.

\*

\* @warning You must not return `nil` from this method.

\*/

- (UIImage \*)avatarPlaceholderImage;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

/\*\*

\* The `JSQMessageBubbleImageDataSource` protocol defines the common interface through which

\* a `JSQMessagesViewController` and `JSQMessagesCollectionView` interact with

\* message bubble image model objects.

\*

\* It declares the required and optional methods that a class must implement so that instances

\* of that class can be display properly within a `JSQMessagesCollectionViewCell`.

\*

\* A concrete class that conforms to this protocol is provided in the library. See `JSQMessagesBubbleImage`.

\*

\* @see JSQMessagesBubbleImage.

\*/

@protocol JSQMessageBubbleImageDataSource <NSObject>

@required

/\*\*

\* @return The message bubble image for a regular display state.

\*

\* @warning You must not return `nil` from this method.

\*/

- (UIImage \*)messageBubbleImage;

/\*\*

\* @return The message bubble image for a highlighted display state.

\*

\* @warning You must not return `nil` from this method.

\*/

- (UIImage \*)messageBubbleHighlightedImage;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import "JSQMessageMediaData.h"

/\*\*

\* The `JSQMessageData` protocol defines the common interface through which

\* a `JSQMessagesViewController` and `JSQMessagesCollectionView` interact with message model objects.

\*

\* It declares the required and optional methods that a class must implement so that instances of that class

\* can be displayed properly within a `JSQMessagesCollectionViewCell`.

\*

\* The class that conforms to this protocol is provided in the library. See `JSQMessage`.

\*

\* @see JSQMessage.

\*/

@protocol JSQMessageData <NSObject>

@required

/\*\*

\* @return A string identifier that uniquely identifies the user who sent the message.

\*

\* @discussion If you need to generate a unique identifier, consider using

\* `[[NSProcessInfo processInfo] globallyUniqueString]`

\*

\* @warning You must not return `nil` from this method. This value must be unique.

\*/

- (NSString \*)senderId;

/\*\*

\* @return The display name for the user who sent the message.

\*

\* @warning You must not return `nil` from this method.

\*/

- (NSString \*)senderDisplayName;

/\*\*

\* @return The date that the message was sent.

\*

\* @warning You must not return `nil` from this method.

\*/

- (NSDate \*)date;

/\*\*

\* This method is used to determine if the message data item contains text or media.

\* If this method returns `YES`, an instance of `JSQMessagesViewController` will ignore

\* the `text` method of this protocol when dequeuing a `JSQMessagesCollectionViewCell`

\* and only call the `media` method.

\*

\* Similarly, if this method returns `NO` then the `media` method will be ignored and

\* and only the `text` method will be called.

\*

\* @return A boolean value specifying whether or not this is a media message or a text message.

\* Return `YES` if this item is a media message, and `NO` if it is a text message.

\*/

- (BOOL)isMediaMessage;

/\*\*

\* @return An integer that can be used as a table address in a hash table structure.

\*

\* @discussion This value must be unique for each message with distinct contents.

\* This value is used to cache layout information in the collection view.

\*/

- (NSUInteger)messageHash;

@optional

/\*\*

\* @return The body text of the message.

\*

\* @warning You must not return `nil` from this method.

\*/

- (NSString \*)text;

/\*\*

\* @return The media item of the message.

\*

\* @warning You must not return `nil` from this method.

\*/

- (id<JSQMessageMediaData>)media;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

/\*\*

\* The `JSQMessageMediaData` protocol defines the common interface through which

\* a `JSQMessagesViewController` and `JSQMessagesCollectionView` interact with media message model objects.

\*

\* It declares the required and optional methods that a class must implement so that instances of that class

\* can be displayed properly within a `JSQMessagesCollectionViewCell`.

\*

\* This library provides a few concrete classes that conform to this protocol. You may use them as-is,

\* but they will likely require some modifications or extensions to conform to your particular data models.

\* These concrete media items are: `JSQPhotoMediaItem`, `JSQLocationMediaItem`, `JSQVideoMediaItem`.

\*

\* @see JSQPhotoMediaItem.

\* @see JSQLocationMediaItem.

\* @see JSQVideoMediaItem.

\*/

@protocol JSQMessageMediaData <NSObject>

@required

/\*\*

\* @return An initialized `UIView` object that represents the data for this media object.

\*

\* @discussion You may return `nil` from this method while the media data is being downloaded.

\*/

- (UIView \*)mediaView;

/\*\*

\* @return The frame size for the mediaView when displayed in a `JSQMessagesCollectionViewCell`.

\*

\* @discussion You should return an appropriate size value to be set for the mediaView's frame

\* based on the contents of the view, and the frame and layout of the `JSQMessagesCollectionViewCell`

\* in which mediaView will be displayed.

\*

\* @warning You must return a size with non-zero, positive width and height values.

\*/

- (CGSize)mediaViewDisplaySize;

/\*\*

\* @return A placeholder media view to be displayed if mediaView is not yet available, or `nil`.

\* For example, if mediaView will be constructed based on media data that must be downloaded,

\* this placeholder view will be used until mediaView is not `nil`.

\*

\* @discussion If you do not need support for a placeholder view, then you may simply return the

\* same value here as mediaView. Otherwise, consider using `JSQMessagesMediaPlaceholderView`.

\*

\* @warning You must not return `nil` from this method.

\*

\* @see JSQMessagesMediaPlaceholderView.

\*/

- (UIView \*)mediaPlaceholderView;

/\*\*

\* @return An integer that can be used as a table address in a hash table structure.

\*

\* @discussion This value must be unique for each media item with distinct contents.

\* This value is used to cache layout information in the collection view.

\*/

- (NSUInteger)mediaHash;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

#import "JSQMessageAvatarImageDataSource.h"

/\*\*

\* A `JSQMessagesAvatarImage` model object represents an avatar image.

\* This is a concrete class that implements the `JSQMessageAvatarImageDataSource` protocol.

\* It contains a regular avatar image, a highlighted avatar image, and a placeholder avatar image.

\*

\* @see JSQMessagesAvatarImageFactory.

\*/

@interface JSQMessagesAvatarImage : NSObject <JSQMessageAvatarImageDataSource, NSCopying>

/\*\*

\* The avatar image for a regular display state.

\*/

@property (nonatomic, strong) UIImage \*avatarImage;

/\*\*

\* The avatar image for a highlighted display state.

\*/

@property (nonatomic, strong) UIImage \*avatarHighlightedImage;

/\*\*

\* Returns the placeholder image for an avatar to display if avatarImage is `nil`.

\*/

@property (nonatomic, strong, readonly) UIImage \*avatarPlaceholderImage;

/\*\*

\* Initializes and returns an avatar image object having the specified image.

\*

\* @param image The image for this avatar image. This image will be used for the all of the following

\* properties: avatarImage, avatarHighlightedImage, avatarPlaceholderImage;

\* This value must not be `nil`.

\*

\* @return An initialized `JSQMessagesAvatarImage` object if successful, `nil` otherwise.

\*/

+ (instancetype)avatarWithImage:(UIImage \*)image;

/\*\*

\* Initializes and returns an avatar image object having the specified placeholder image.

\*

\* @param placeholderImage The placeholder image for this avatar image. This value must not be `nil`.

\*

\* @return An initialized `JSQMessagesAvatarImage` object if successful, `nil` otherwise.

\*/

+ (instancetype)avatarImageWithPlaceholder:(UIImage \*)placeholderImage;

/\*\*

\* Initializes and returns an avatar image object having the specified regular, highlighed, and placeholder images.

\*

\* @param avatarImage The avatar image for a regular display state.

\* @param highlightedImage The avatar image for a highlighted display state.

\* @param placeholderImage The placeholder image for this avatar image. This value must not be `nil`.

\*

\* @return An initialized `JSQMessagesAvatarImage` object if successful, `nil` otherwise.

\*/

- (instancetype)initWithAvatarImage:(UIImage \*)avatarImage

highlightedImage:(UIImage \*)highlightedImage

placeholderImage:(UIImage \*)placeholderImage NS\_DESIGNATED\_INITIALIZER;

/\*\*

\* Not a valid initializer.

\*/

- (id)init NS\_UNAVAILABLE;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

#import "JSQMessageBubbleImageDataSource.h"

/\*\*

\* A `JSQMessagesBubbleImage` model object represents a message bubble image, and is immutable.

\* This is a concrete class that implements the `JSQMessageBubbleImageDataSource` protocol.

\* It contains a regular message bubble image and a highlighted message bubble image.

\*

\* @see JSQMessagesBubbleImageFactory.

\*/

@interface JSQMessagesBubbleImage : NSObject <JSQMessageBubbleImageDataSource, NSCopying>

/\*\*

\* Returns the message bubble image for a regular display state.

\*/

@property (strong, nonatomic, readonly) UIImage \*messageBubbleImage;

/\*\*

\* Returns the message bubble image for a highlighted display state.

\*/

@property (strong, nonatomic, readonly) UIImage \*messageBubbleHighlightedImage;

/\*\*

\* Initializes and returns a message bubble image object having the specified regular image and highlighted image.

\*

\* @param image The regular message bubble image. This value must not be `nil`.

\* @param highlightedImage The highlighted message bubble image. This value must not be `nil`.

\*

\* @return An initialized `JSQMessagesBubbleImage` object if successful, `nil` otherwise.

\*

\* @see JSQMessagesBubbleImageFactory.

\*/

- (instancetype)initWithMessageBubbleImage:(UIImage \*)image highlightedImage:(UIImage \*)highlightedImage NS\_DESIGNATED\_INITIALIZER;

/\*\*

\* Not a valid initializer.

\*/

- (id)init NS\_UNAVAILABLE;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

@class JSQMessagesCollectionView;

@protocol JSQMessageData;

@protocol JSQMessageBubbleImageDataSource;

@protocol JSQMessageAvatarImageDataSource;

/\*\*

\* An object that adopts the `JSQMessagesCollectionViewDataSource` protocol is responsible for providing the data and views

\* required by a `JSQMessagesCollectionView`. The data source object represents your app’s messaging data model

\* and vends information to the collection view as needed.

\*/

@protocol JSQMessagesCollectionViewDataSource <UICollectionViewDataSource>

@required

/\*\*

\* Asks the data source for the current sender's display name, that is, the current user who is sending messages.

\*

\* @return An initialized string describing the current sender to display in a `JSQMessagesCollectionViewCell`.

\*

\* @warning You must not return `nil` from this method. This value does not need to be unique.

\*/

- (NSString \*)senderDisplayName;

/\*\*

\* Asks the data source for the current sender's unique identifier, that is, the current user who is sending messages.

\*

\* @return An initialized string identifier that uniquely identifies the current sender.

\*

\* @warning You must not return `nil` from this method. This value must be unique.

\*/

- (NSString \*)senderId;

/\*\*

\* Asks the data source for the message data that corresponds to the specified item at indexPath in the collectionView.

\*

\* @param collectionView The collection view requesting this information.

\* @param indexPath The index path that specifies the location of the item.

\*

\* @return An initialized object that conforms to the `JSQMessageData` protocol. You must not return `nil` from this method.

\*/

- (id<JSQMessageData>)collectionView:(JSQMessagesCollectionView \*)collectionView messageDataForItemAtIndexPath:(NSIndexPath \*)indexPath;

/\*\*

\* Notifies the data source that the item at indexPath has been deleted.

\* Implementations of this method should remove the item from the data source.

\*

\* @param collectionView The collection view requesting this information.

\* @param indexPath The index path that specifies the location of the item.

\*/

- (void)collectionView:(JSQMessagesCollectionView \*)collectionView didDeleteMessageAtIndexPath:(NSIndexPath \*)indexPath;

/\*\*

\* Asks the data source for the message bubble image data that corresponds to the specified message data item at indexPath in the collectionView.

\*

\* @param collectionView The collection view requesting this information.

\* @param indexPath The index path that specifies the location of the item.

\*

\* @return An initialized object that conforms to the `JSQMessageBubbleImageDataSource` protocol. You may return `nil` from this method if you do not

\* want the specified item to display a message bubble image.

\*

\* @discussion It is recommended that you utilize `JSQMessagesBubbleImageFactory` to return valid `JSQMessagesBubbleImage` objects.

\* However, you may provide your own data source object as long as it conforms to the `JSQMessageBubbleImageDataSource` protocol.

\*

\* @warning Note that providing your own bubble image data source objects may require additional

\* configuration of the collectionView layout object, specifically regarding its `messageBubbleTextViewFrameInsets` and `messageBubbleTextViewTextContainerInsets`.

\*

\* @see JSQMessagesBubbleImageFactory.

\* @see JSQMessagesCollectionViewFlowLayout.

\*/

- (id<JSQMessageBubbleImageDataSource>)collectionView:(JSQMessagesCollectionView \*)collectionView messageBubbleImageDataForItemAtIndexPath:(NSIndexPath \*)indexPath;

/\*\*

\* Asks the data source for the avatar image data that corresponds to the specified message data item at indexPath in the collectionView.

\*

\* @param collectionView The collection view requesting this information.

\* @param indexPath The index path that specifies the location of the item.

\*

\* @return A initialized object that conforms to the `JSQMessageAvatarImageDataSource` protocol. You may return `nil` from this method if you do not want

\* the specified item to display an avatar.

\*

\* @discussion It is recommended that you utilize `JSQMessagesAvatarImageFactory` to return valid `JSQMessagesAvatarImage` objects.

\* However, you may provide your own data source object as long as it conforms to the `JSQMessageAvatarImageDataSource` protocol.

\*

\* @see JSQMessagesAvatarImageFactory.

\* @see JSQMessagesCollectionViewFlowLayout.

\*/

- (id<JSQMessageAvatarImageDataSource>)collectionView:(JSQMessagesCollectionView \*)collectionView avatarImageDataForItemAtIndexPath:(NSIndexPath \*)indexPath;

@optional

/\*\*

\* Asks the data source for the text to display in the `cellTopLabel` for the specified

\* message data item at indexPath in the collectionView.

\*

\* @param collectionView The collection view requesting this information.

\* @param indexPath The index path that specifies the location of the item.

\*

\* @return A configured attributed string or `nil` if you do not want text displayed for the item at indexPath.

\* Return an attributed string with `nil` attributes to use the default attributes.

\*

\* @see JSQMessagesCollectionViewCell.

\*/

- (NSAttributedString \*)collectionView:(JSQMessagesCollectionView \*)collectionView attributedTextForCellTopLabelAtIndexPath:(NSIndexPath \*)indexPath;

/\*\*

\* Asks the data source for the text to display in the `messageBubbleTopLabel` for the specified

\* message data item at indexPath in the collectionView.

\*

\* @param collectionView The collection view requesting this information.

\* @param indexPath The index path that specifies the location of the item.

\*

\* @return A configured attributed string or `nil` if you do not want text displayed for the item at indexPath.

\* Return an attributed string with `nil` attributes to use the default attributes.

\*

\* @see JSQMessagesCollectionViewCell.

\*/

- (NSAttributedString \*)collectionView:(JSQMessagesCollectionView \*)collectionView attributedTextForMessageBubbleTopLabelAtIndexPath:(NSIndexPath \*)indexPath;

/\*\*

\* Asks the data source for the text to display in the `cellBottomLabel` for the the specified

\* message data item at indexPath in the collectionView.

\*

\* @param collectionView The collection view requesting this information.

\* @param indexPath The index path that specifies the location of the item.

\*

\* @return A configured attributed string or `nil` if you do not want text displayed for the item at indexPath.

\* Return an attributed string with `nil` attributes to use the default attributes.

\*

\* @see JSQMessagesCollectionViewCell.

\*/

- (NSAttributedString \*)collectionView:(JSQMessagesCollectionView \*)collectionView attributedTextForCellBottomLabelAtIndexPath:(NSIndexPath \*)indexPath;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

@class JSQMessagesCollectionView;

@class JSQMessagesCollectionViewFlowLayout;

@class JSQMessagesCollectionViewCell;

@class JSQMessagesLoadEarlierHeaderView;

/\*\*

\* The `JSQMessagesCollectionViewDelegateFlowLayout` protocol defines methods that allow you to

\* manage additional layout information for the collection view and respond to additional actions on its items.

\* The methods of this protocol are all optional.

\*/

@protocol JSQMessagesCollectionViewDelegateFlowLayout <UICollectionViewDelegateFlowLayout>

@optional

/\*\*

\* Asks the delegate for the height of the `cellTopLabel` for the item at the specified indexPath.

\*

\* @param collectionView The collection view object displaying the flow layout.

\* @param collectionViewLayout The layout object requesting the information.

\* @param indexPath The index path of the item.

\*

\* @return The height of the `cellTopLabel` for the item at indexPath.

\*

\* @see JSQMessagesCollectionViewCell.

\*/

- (CGFloat)collectionView:(JSQMessagesCollectionView \*)collectionView

layout:(JSQMessagesCollectionViewFlowLayout \*)collectionViewLayout heightForCellTopLabelAtIndexPath:(NSIndexPath \*)indexPath;

/\*\*

\* Asks the delegate for the height of the `messageBubbleTopLabel` for the item at the specified indexPath.

\*

\* @param collectionView The collection view object displaying the flow layout.

\* @param collectionViewLayout The layout object requesting the information.

\* @param indexPath The index path of the item.

\*

\* @return The height of the `messageBubbleTopLabel` for the item at indexPath.

\*

\* @see JSQMessagesCollectionViewCell.

\*/

- (CGFloat)collectionView:(JSQMessagesCollectionView \*)collectionView

layout:(JSQMessagesCollectionViewFlowLayout \*)collectionViewLayout heightForMessageBubbleTopLabelAtIndexPath:(NSIndexPath \*)indexPath;

/\*\*

\* Asks the delegate for the height of the `cellBottomLabel` for the item at the specified indexPath.

\*

\* @param collectionView The collection view object displaying the flow layout.

\* @param collectionViewLayout The layout object requesting the information.

\* @param indexPath The index path of the item.

\*

\* @return The height of the `cellBottomLabel` for the item at indexPath.

\*

\* @see JSQMessagesCollectionViewCell.

\*/

- (CGFloat)collectionView:(JSQMessagesCollectionView \*)collectionView

layout:(JSQMessagesCollectionViewFlowLayout \*)collectionViewLayout heightForCellBottomLabelAtIndexPath:(NSIndexPath \*)indexPath;

/\*\*

\* Notifies the delegate that the avatar image view at the specified indexPath did receive a tap event.

\*

\* @param collectionView The collection view object that is notifying the delegate of the tap event.

\* @param avatarImageView The avatar image view that was tapped.

\* @param indexPath The index path of the item for which the avatar was tapped.

\*/

- (void)collectionView:(JSQMessagesCollectionView \*)collectionView didTapAvatarImageView:(UIImageView \*)avatarImageView atIndexPath:(NSIndexPath \*)indexPath;

/\*\*

\* Notifies the delegate that the message bubble at the specified indexPath did receive a tap event.

\*

\* @param collectionView The collection view object that is notifying the delegate of the tap event.

\* @param indexPath The index path of the item for which the message bubble was tapped.

\*/

- (void)collectionView:(JSQMessagesCollectionView \*)collectionView didTapMessageBubbleAtIndexPath:(NSIndexPath \*)indexPath;

/\*\*

\* Notifies the delegate that the cell at the specified indexPath did receive a tap event at the specified touchLocation.

\*

\* @param collectionView The collection view object that is notifying the delegate of the tap event.

\* @param indexPath The index path of the item for which the message bubble was tapped.

\* @param touchLocation The location of the touch event in the cell's coordinate system.

\*

\* @warning This method is \*only\* called if position is \*not\* within the bounds of the cell's

\* avatar image view or message bubble image view. In other words, this method is \*not\* called when the cell's

\* avatar or message bubble are tapped. There are separate delegate methods for these two cases.

\*

\* @see `collectionView:didTapAvatarImageView:atIndexPath:`

\* @see `collectionView:didTapMessageBubbleAtIndexPath:atIndexPath:`

\*/

- (void)collectionView:(JSQMessagesCollectionView \*)collectionView didTapCellAtIndexPath:(NSIndexPath \*)indexPath touchLocation:(CGPoint)touchLocation;

/\*\*

\* Notifies the delegate that the collection view's header did receive a tap event.

\*

\* @param collectionView The collection view object that is notifying the delegate of the tap event.

\* @param headerView The header view in the collection view.

\* @param sender The button that was tapped.

\*/

- (void)collectionView:(JSQMessagesCollectionView \*)collectionView

header:(JSQMessagesLoadEarlierHeaderView \*)headerView didTapLoadEarlierMessagesButton:(UIButton \*)sender;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import "JSQMediaItem.h"

/\*\*

\* The `JSQPhotoMediaItem` class is a concrete `JSQMediaItem` subclass that implements the `JSQMessageMediaData` protocol

\* and represents a photo media message. An initialized `JSQPhotoMediaItem` object can be passed

\* to a `JSQMediaMessage` object during its initialization to construct a valid media message object.

\* You may wish to subclass `JSQPhotoMediaItem` to provide additional functionality or behavior.

\*/

@interface JSQPhotoMediaItem : JSQMediaItem <JSQMessageMediaData, NSCoding, NSCopying>

/\*\*

\* The image for the photo media item. The default value is `nil`.

\*/

@property (copy, nonatomic) UIImage \*image;

/\*\*

\* Initializes and returns a photo media item object having the given image.

\*

\* @param image The image for the photo media item. This value may be `nil`.

\*

\* @return An initialized `JSQPhotoMediaItem` if successful, `nil` otherwise.

\*

\* @discussion If the image must be dowloaded from the network,

\* you may initialize a `JSQPhotoMediaItem` object with a `nil` image.

\* Once the image has been retrieved, you can then set the image property.

\*/

- (instancetype)initWithImage:(UIImage \*)image;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import "JSQMediaItem.h"

/\*\*

\* The `JSQVideoMediaItem` class is a concrete `JSQMediaItem` subclass that implements the `JSQMessageMediaData` protocol

\* and represents a video media message. An initialized `JSQVideoMediaItem` object can be passed

\* to a `JSQMediaMessage` object during its initialization to construct a valid media message object.

\* You may wish to subclass `JSQVideoMediaItem` to provide additional functionality or behavior.

\*/

@interface JSQVideoMediaItem : JSQMediaItem <JSQMessageMediaData, NSCoding, NSCopying>

/\*\*

\* The URL that identifies a video resource.

\*/

@property (nonatomic, strong) NSURL \*fileURL;

/\*\*

\* A boolean value that specifies whether or not the video is ready to be played.

\*

\* @discussion When set to `YES`, the video is ready. When set to `NO` it is not ready.

\*/

@property (nonatomic, assign) BOOL isReadyToPlay;

/\*\*

\* Initializes and returns a video media item having the given fileURL.

\*

\* @param fileURL The URL that identifies the video resource.

\* @param isReadyToPlay A boolean value that specifies if the video is ready to play.

\*

\* @return An initialized `JSQVideoMediaItem` if successful, `nil` otherwise.

\*

\* @discussion If the video must be downloaded from the network,

\* you may initialize a `JSQVideoMediaItem` with a `nil` fileURL or specify `NO` for

\* isReadyToPlay. Once the video has been saved to disk, or is ready to stream, you can

\* set the fileURL property or isReadyToPlay property, respectively.

\*/

- (instancetype)initWithFileURL:(NSURL \*)fileURL isReadyToPlay:(BOOL)isReadyToPlay;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <UIKit/UIKit.h>

/\*\*

\* `JSQMessagesCellTextView` is a subclass of `UITextView` that is used to display text

\* in a `JSQMessagesCollectionViewCell`.

\*/

@interface JSQMessagesCellTextView : UITextView

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <UIKit/UIKit.h>

#import "JSQMessagesCollectionViewFlowLayout.h"

#import "JSQMessagesCollectionViewDelegateFlowLayout.h"

#import "JSQMessagesCollectionViewDataSource.h"

#import "JSQMessagesCollectionViewCell.h"

@class JSQMessagesTypingIndicatorFooterView;

@class JSQMessagesLoadEarlierHeaderView;

/\*\*

\* The `JSQMessagesCollectionView` class manages an ordered collection of message data items and presents

\* them using a specialized layout for messages.

\*/

@interface JSQMessagesCollectionView : UICollectionView <JSQMessagesCollectionViewCellDelegate>

/\*\*

\* The object that provides the data for the collection view.

\* The data source must adopt the `JSQMessagesCollectionViewDataSource` protocol.

\*/

@property (weak, nonatomic) id<JSQMessagesCollectionViewDataSource> dataSource;

/\*\*

\* The object that acts as the delegate of the collection view.

\* The delegate must adopt the `JSQMessagesCollectionViewDelegateFlowLayout` protocol.

\*/

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

/\*\*

\* The layout used to organize the collection view’s items.

\*/

@property (strong, nonatomic) JSQMessagesCollectionViewFlowLayout \*collectionViewLayout;

/\*\*

\* Specifies whether the typing indicator displays on the left or right side of the collection view

\* when shown. That is, whether it displays for an "incoming" or "outgoing" message.

\* The default value is `YES`, meaning that the typing indicator will display on the left side of the

\* collection view for incoming messages.

\*

\* @discussion If your `JSQMessagesViewController` subclass displays messages for right-to-left

\* languages, such as Arabic, set this property to `NO`.

\*

\*/

@property (assign, nonatomic) BOOL typingIndicatorDisplaysOnLeft;

/\*\*

\* The color of the typing indicator message bubble. The default value is a light gray color.

\*/

@property (strong, nonatomic) UIColor \*typingIndicatorMessageBubbleColor;

/\*\*

\* The color of the typing indicator ellipsis. The default value is a dark gray color.

\*/

@property (strong, nonatomic) UIColor \*typingIndicatorEllipsisColor;

/\*\*

\* The color of the text in the load earlier messages header. The default value is a bright blue color.

\*/

@property (strong, nonatomic) UIColor \*loadEarlierMessagesHeaderTextColor;

/\*\*

\* Returns a `JSQMessagesTypingIndicatorFooterView` object for the specified index path

\* that is configured using the collection view's properties:

\* typingIndicatorDisplaysOnLeft, typingIndicatorMessageBubbleColor, typingIndicatorEllipsisColor.

\*

\* @param indexPath The index path specifying the location of the supplementary view in the collection view. This value must not be `nil`.

\*

\* @return A valid `JSQMessagesTypingIndicatorFooterView` object.

\*/

- (JSQMessagesTypingIndicatorFooterView \*)dequeueTypingIndicatorFooterViewForIndexPath:(NSIndexPath \*)indexPath;

/\*\*

\* Returns a `JSQMessagesLoadEarlierHeaderView` object for the specified index path

\* that is configured using the collection view's loadEarlierMessagesHeaderTextColor property.

\*

\* @param indexPath The index path specifying the location of the supplementary view in the collection view. This value must not be `nil`.

\*

\* @return A valid `JSQMessagesLoadEarlierHeaderView` object.

\*/

- (JSQMessagesLoadEarlierHeaderView \*)dequeueLoadEarlierMessagesViewHeaderForIndexPath:(NSIndexPath \*)indexPath;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <UIKit/UIKit.h>

#import "JSQMessagesLabel.h"

#import "JSQMessagesCellTextView.h"

@class JSQMessagesCollectionViewCell;

/\*\*

\* The `JSQMessagesCollectionViewCellDelegate` protocol defines methods that allow you to manage

\* additional interactions within the collection view cell.

\*/

@protocol JSQMessagesCollectionViewCellDelegate <NSObject>

@required

/\*\*

\* Tells the delegate that the avatarImageView of the cell has been tapped.

\*

\* @param cell The cell that received the tap touch event.

\*/

- (void)messagesCollectionViewCellDidTapAvatar:(JSQMessagesCollectionViewCell \*)cell;

/\*\*

\* Tells the delegate that the message bubble of the cell has been tapped.

\*

\* @param cell The cell that received the tap touch event.

\*/

- (void)messagesCollectionViewCellDidTapMessageBubble:(JSQMessagesCollectionViewCell \*)cell;

/\*\*

\* Tells the delegate that the cell has been tapped at the point specified by position.

\*

\* @param cell The cell that received the tap touch event.

\* @param position The location of the received touch in the cell's coordinate system.

\*

\* @discussion This method is \*only\* called if position is \*not\* within the bounds of the cell's

\* avatar image view or message bubble image view. In other words, this method is \*not\* called when the cell's

\* avatar or message bubble are tapped.

\*

\* @see `messagesCollectionViewCellDidTapAvatar:`

\* @see `messagesCollectionViewCellDidTapMessageBubble:`

\*/

- (void)messagesCollectionViewCellDidTapCell:(JSQMessagesCollectionViewCell \*)cell atPosition:(CGPoint)position;

/\*\*

\* Tells the delegate that an actions has been selected from the menu of this cell.

\* This method is automatically called for any registered actions.

\*

\* @param cell The cell that displayed the menu.

\* @param action The action that has been performed.

\* @param sender The object that initiated the action.

\*

\* @see `JSQMessagesCollectionViewCell`

\*/

- (void)messagesCollectionViewCell:(JSQMessagesCollectionViewCell \*)cell didPerformAction:(SEL)action withSender:(id)sender;

@end

/\*\*

\* The `JSQMessagesCollectionViewCell` is an abstract base class that presents the content for

\* a single message data item when that item is within the collection view’s visible bounds.

\* The layout and presentation of cells is managed by the collection view and its corresponding layout object.

\*

\* @warning This class is intended to be subclassed. You should not use it directly.

\*

\* @see JSQMessagesCollectionViewCellIncoming.

\* @see JSQMessagesCollectionViewCellOutgoing.

\*/

@interface JSQMessagesCollectionViewCell : UICollectionViewCell

/\*\*

\* The object that acts as the delegate for the cell.

\*/

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

/\*\*

\* Returns the label that is pinned to the top of the cell.

\* This label is most commonly used to display message timestamps.

\*/

@property (weak, nonatomic, readonly) JSQMessagesLabel \*cellTopLabel;

/\*\*

\* Returns the label that is pinned just above the messageBubbleImageView, and below the cellTopLabel.

\* This label is most commonly used to display the message sender.

\*/

@property (weak, nonatomic, readonly) JSQMessagesLabel \*messageBubbleTopLabel;

/\*\*

\* Returns the label that is pinned to the bottom of the cell.

\* This label is most commonly used to display message delivery status.

\*/

@property (weak, nonatomic, readonly) JSQMessagesLabel \*cellBottomLabel;

/\*\*

\* Returns the text view of the cell. This text view contains the message body text.

\*

\* @warning If mediaView returns a non-nil view, then this value will be `nil`.

\*/

@property (weak, nonatomic, readonly) JSQMessagesCellTextView \*textView;

/\*\*

\* Returns the bubble image view of the cell that is responsible for displaying message bubble images.

\*

\* @warning If mediaView returns a non-nil view, then this value will be `nil`.

\*/

@property (weak, nonatomic, readonly) UIImageView \*messageBubbleImageView;

/\*\*

\* Returns the message bubble container view of the cell. This view is the superview of

\* the cell's textView and messageBubbleImageView.

\*

\* @discussion You may customize the cell by adding custom views to this container view.

\* To do so, override `collectionView:cellForItemAtIndexPath:`

\*

\* @warning You should not try to manipulate any properties of this view, for example adjusting

\* its frame, nor should you remove this view from the cell or remove any of its subviews.

\* Doing so could result in unexpected behavior.

\*/

@property (weak, nonatomic, readonly) UIView \*messageBubbleContainerView;

/\*\*

\* Returns the avatar image view of the cell that is responsible for displaying avatar images.

\*/

@property (weak, nonatomic, readonly) UIImageView \*avatarImageView;

/\*\*

\* Returns the avatar container view of the cell. This view is the superview of the cell's avatarImageView.

\*

\* @discussion You may customize the cell by adding custom views to this container view.

\* To do so, override `collectionView:cellForItemAtIndexPath:`

\*

\* @warning You should not try to manipulate any properties of this view, for example adjusting

\* its frame, nor should you remove this view from the cell or remove any of its subviews.

\* Doing so could result in unexpected behavior.

\*/

@property (weak, nonatomic, readonly) UIView \*avatarContainerView;

/\*\*

\* The media view of the cell. This view displays the contents of a media message.

\*

\* @warning If this value is non-nil, then textView and messageBubbleImageView will both be `nil`.

\*/

@property (weak, nonatomic) UIView \*mediaView;

/\*\*

\* Returns the underlying gesture recognizer for tap gestures in the avatarImageView of the cell.

\* This gesture handles the tap event for the avatarImageView and notifies the cell's delegate.

\*/

@property (weak, nonatomic, readonly) UITapGestureRecognizer \*tapGestureRecognizer;

#pragma mark - Class methods

/\*\*

\* Returns the `UINib` object initialized for the cell.

\*

\* @return The initialized `UINib` object or `nil` if there were errors during

\* initialization or the nib file could not be located.

\*/

+ (UINib \*)nib;

/\*\*

\* Returns the default string used to identify a reusable cell for text message items.

\*

\* @return The string used to identify a reusable cell.

\*/

+ (NSString \*)cellReuseIdentifier;

/\*\*

\* Returns the default string used to identify a reusable cell for media message items.

\*

\* @return The string used to identify a reusable cell.

\*/

+ (NSString \*)mediaCellReuseIdentifier;

/\*\*

\* Registers an action to be available in the cell's menu.

\*

\* @param action The selector to register with the cell.

\*

\* @discussion Non-standard or non-system actions must be added to the `UIMenuController` manually.

\* You can do this by creating a new `UIMenuItem` and adding it via the controller's `menuItems` property.

\*

\* @warning Note that all message cells share the all actions registered here.

\*/

+ (void)registerMenuAction:(SEL)action;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import "JSQMessagesCollectionViewCell.h"

/\*\*

\* A `JSQMessagesCollectionViewCellIncoming` object is a concrete instance

\* of `JSQMessagesCollectionViewCell` that represents an incoming message data item.

\*/

@interface JSQMessagesCollectionViewCellIncoming : JSQMessagesCollectionViewCell

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import "JSQMessagesCollectionViewCell.h"

/\*\*

\* A `JSQMessagesCollectionViewCellOutgoing` object is a concrete instance

\* of `JSQMessagesCollectionViewCell` that represents an outgoing message data item.

\*/

@interface JSQMessagesCollectionViewCellOutgoing : JSQMessagesCollectionViewCell

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <UIKit/UIKit.h>

@class JSQMessagesComposerTextView;

/\*\*

\* A delegate object used to notify the receiver of paste events from a `JSQMessagesComposerTextView`.

\*/

@protocol JSQMessagesComposerTextViewPasteDelegate <NSObject>

/\*\*

\* Asks the delegate whether or not the `textView` should use the original implementation of `-[UITextView paste]`.

\*

\* @discussion Use this delegate method to implement custom pasting behavior.

\* You should return `NO` when you want to handle pasting.

\* Return `YES` to defer functionality to the `textView`.

\*/

- (BOOL)composerTextView:(JSQMessagesComposerTextView \*)textView shouldPasteWithSender:(id)sender;

@end

/\*\*

\* An instance of `JSQMessagesComposerTextView` is a subclass of `UITextView` that is styled and used

\* for composing messages in a `JSQMessagesViewController`. It is a subview of a `JSQMessagesToolbarContentView`.

\*/

@interface JSQMessagesComposerTextView : UITextView

/\*\*

\* The text to be displayed when the text view is empty. The default value is `nil`.

\*/

@property (copy, nonatomic) NSString \*placeHolder;

/\*\*

\* The color of the place holder text. The default value is `[UIColor lightGrayColor]`.

\*/

@property (strong, nonatomic) UIColor \*placeHolderTextColor;

/\*\*

\* The object that acts as the paste delegate of the text view.

\*/

@property (weak, nonatomic) id<JSQMessagesComposerTextViewPasteDelegate> pasteDelegate;

/\*\*

\* Determines whether or not the text view contains text after trimming white space

\* from the front and back of its string.

\*

\* @return `YES` if the text view contains text, `NO` otherwise.

\*/

- (BOOL)hasText;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

#import "JSQMessagesToolbarContentView.h"

@class JSQMessagesInputToolbar;

/\*\*

\* The `JSQMessagesInputToolbarDelegate` protocol defines methods for interacting with

\* a `JSQMessagesInputToolbar` object.

\*/

@protocol JSQMessagesInputToolbarDelegate <UIToolbarDelegate>

@required

/\*\*

\* Tells the delegate that the toolbar's `rightBarButtonItem` has been pressed.

\*

\* @param toolbar The object representing the toolbar sending this information.

\* @param sender The button that received the touch event.

\*/

- (void)messagesInputToolbar:(JSQMessagesInputToolbar \*)toolbar

didPressRightBarButton:(UIButton \*)sender;

/\*\*

\* Tells the delegate that the toolbar's `leftBarButtonItem` has been pressed.

\*

\* @param toolbar The object representing the toolbar sending this information.

\* @param sender The button that received the touch event.

\*/

- (void)messagesInputToolbar:(JSQMessagesInputToolbar \*)toolbar

didPressLeftBarButton:(UIButton \*)sender;

@end

/\*\*

\* An instance of `JSQMessagesInputToolbar` defines the input toolbar for

\* composing a new message. It is displayed above and follow the movement of the system keyboard.

\*/

@interface JSQMessagesInputToolbar : UIToolbar

/\*\*

\* The object that acts as the delegate of the toolbar.

\*/

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

/\*\*

\* Returns the content view of the toolbar. This view contains all subviews of the toolbar.

\*/

@property (weak, nonatomic, readonly) JSQMessagesToolbarContentView \*contentView;

/\*\*

\* A boolean value indicating whether the send button is on the right side of the toolbar or not.

\*

\* @discussion The default value is `YES`, which indicates that the send button is the right-most subview of

\* the toolbar's `contentView`. Set to `NO` to specify that the send button is on the left. This

\* property is used to determine which touch events correspond to which actions.

\*

\* @warning Note, this property \*does not\* change the positions of buttons in the toolbar's content view.

\* It only specifies whether the `rightBarButtonItem `or the `leftBarButtonItem` is the send button.

\* The other button then acts as the accessory button.

\*/

@property (assign, nonatomic) BOOL sendButtonOnRight;

/\*\*

\* Specifies the default (minimum) height for the toolbar. The default value is `44.0f`. This value must be positive.

\*/

@property (assign, nonatomic) CGFloat preferredDefaultHeight;

/\*\*

\* Specifies the maximum height for the toolbar. The default value is `NSNotFound`, which specifies no maximum height.

\*/

@property (assign, nonatomic) NSUInteger maximumHeight;

/\*\*

\* Enables or disables the send button based on whether or not its `textView` has text.

\* That is, the send button will be enabled if there is text in the `textView`, and disabled otherwise.

\*/

- (void)toggleSendButtonEnabled;

/\*\*

\* Loads the content view for the toolbar.

\*

\* @discussion Override this method to provide a custom content view for the toolbar.

\*

\* @return An initialized `JSQMessagesToolbarContentView` if successful, otherwise `nil`.

\*/

- (JSQMessagesToolbarContentView \*)loadToolbarContentView;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <UIKit/UIKit.h>

/\*\*

\* `JSQMessagesLabel` is a subclass of `UILabel` that adds support for a `textInsets` property,

\* which is similar to the `textContainerInset` property of `UITextView`.

\*/

@interface JSQMessagesLabel : UILabel

/\*\*

\* The inset of the text layout area within the label's content area. The default value is `UIEdgeInsetsZero`.

\*

\* @discussion This property provides text margins for the text laid out in the label.

\* The inset values provided must be greater than or equal to `0.0f`.

\*/

@property (assign, nonatomic) UIEdgeInsets textInsets;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <UIKit/UIKit.h>

@class JSQMessagesLoadEarlierHeaderView;

/\*\*

\* A constant defining the default height of a `JSQMessagesLoadEarlierHeaderView`.

\*/

FOUNDATION\_EXPORT const CGFloat kJSQMessagesLoadEarlierHeaderViewHeight;

/\*\*

\* The `JSQMessagesLoadEarlierHeaderViewDelegate` defines methods that allow you to

\* respond to interactions within the header view.

\*/

@protocol JSQMessagesLoadEarlierHeaderViewDelegate <NSObject>

@required

/\*\*

\* Tells the delegate that the loadButton has received a touch event.

\*

\* @param headerView The header view that contains the sender.

\* @param sender The button that received the touch.

\*/

- (void)headerView:(JSQMessagesLoadEarlierHeaderView \*)headerView didPressLoadButton:(UIButton \*)sender;

@end

/\*\*

\* The `JSQMessagesLoadEarlierHeaderView` class implements a reusable view that can be placed

\* at the top of a `JSQMessagesCollectionView`. This view contains a "load earlier messages" button

\* and can be used as a way for the user to load previously sent messages.

\*/

@interface JSQMessagesLoadEarlierHeaderView : UICollectionReusableView

/\*\*

\* The object that acts as the delegate of the header view.

\*/

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

/\*\*

\* Returns the load button of the header view.

\*/

@property (weak, nonatomic, readonly) UIButton \*loadButton;

#pragma mark - Class methods

/\*\*

\* Returns the `UINib` object initialized for the collection reusable view.

\*

\* @return The initialized `UINib` object or `nil` if there were errors during

\* initialization or the nib file could not be located.

\*/

+ (UINib \*)nib;

/\*\*

\* Returns the default string used to identify the reusable header view.

\*

\* @return The string used to identify the reusable header view.

\*/

+ (NSString \*)headerReuseIdentifier;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

/\*\*

\* A `JSQMessagesMediaPlaceholderView` object represents a loading or placeholder

\* view for media message objects whose media attachments are not yet available.

\* When sending or receiving media messages that must be uploaded or downloaded from the network,

\* you may display this view temporarily until the media attachement is available.

\* You should return an instance of this class from the `mediaPlaceholderView` method in

\* the `JSQMessageMediaData` protocol.

\*

\* @see JSQMessageMediaData.

\*/

@interface JSQMessagesMediaPlaceholderView : UIView

/\*\*

\* Returns the activity indicator view for this placeholder view, or `nil` if it does not exist.

\*/

@property (nonatomic, weak, readonly) UIActivityIndicatorView \*activityIndicatorView;

/\*\*

\* Returns the image view for this placeholder view, or `nil` if it does not exist.

\*/

@property (nonatomic, weak, readonly) UIImageView \*imageView;

/\*\*

\* Creates a media placeholder view object with a light gray background and

\* a centered activity indicator.

\*

\* @discussion When initializing a `JSQMessagesMediaPlaceholderView` with this method,

\* its imageView property will be nil.

\*

\* @return An initialized `JSQMessagesMediaPlaceholderView` object if successful, `nil` otherwise.

\*/

+ (instancetype)viewWithActivityIndicator;

/\*\*

\* Creates a media placeholder view object with a light gray background and

\* a centered paperclip attachment icon.

\*

\* @discussion When initializing a `JSQMessagesMediaPlaceholderView` with this method,

\* its activityIndicatorView property will be nil.

\*

\* @return An initialized `JSQMessagesMediaPlaceholderView` object if successful, `nil` otherwise.

\*/

+ (instancetype)viewWithAttachmentIcon;

/\*\*

\* Creates a media placeholder view having the given frame, backgroundColor, and activityIndicatorView.

\*

\* @param frame A rectangle defining the frame of the view. This value must be a non-zero, non-null rectangle.

\* @param backgroundColor The background color of the view. This value must not be `nil`.

\* @param activityIndicatorView An initialized activity indicator to be added and centered in the view. This value must not be `nil`.

\*

\* @return An initialized `JSQMessagesMediaPlaceholderView` object if successful, `nil` otherwise.

\*/

- (instancetype)initWithFrame:(CGRect)frame

backgroundColor:(UIColor \*)backgroundColor

activityIndicatorView:(UIActivityIndicatorView \*)activityIndicatorView;

/\*\*

\* Creates a media placeholder view having the given frame, backgroundColor, and imageView.

\*

\* @param frame A rectangle defining the frame of the view. This value must be a non-zero, non-null rectangle.

\* @param backgroundColor The background color of the view. This value must not be `nil`.

\* @param imageView An initialized image view to be added and centered in the view. This value must not be `nil`.

\*

\* @return An initialized `JSQMessagesMediaPlaceholderView` object if successful, `nil` otherwise.

\*/

- (instancetype)initWithFrame:(CGRect)frame

backgroundColor:(UIColor \*)backgroundColor

imageView:(UIImageView \*)imageView;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

#import "JSQMessagesComposerTextView.h"

/\*\*

\* A constant value representing the default spacing to use for the left and right edges

\* of the toolbar content view.

\*/

FOUNDATION\_EXPORT const CGFloat kJSQMessagesToolbarContentViewHorizontalSpacingDefault;

/\*\*

\* A `JSQMessagesToolbarContentView` represents the content displayed in a `JSQMessagesInputToolbar`.

\* These subviews consist of a left button, a text view, and a right button. One button is used as

\* the send button, and the other as the accessory button. The text view is used for composing messages.

\*/

@interface JSQMessagesToolbarContentView : UIView

/\*\*

\* Returns the text view in which the user composes a message.

\*/

@property (weak, nonatomic, readonly) JSQMessagesComposerTextView \*textView;

/\*\*

\* A custom button item displayed on the left of the toolbar content view.

\*

\* @discussion The frame height of this button is ignored. When you set this property, the button

\* is fitted within a pre-defined default content view, the leftBarButtonContainerView,

\* whose height is determined by the height of the toolbar. However, the width of this button

\* will be preserved. You may specify a new width using `leftBarButtonItemWidth`.

\* If the frame of this button is equal to `CGRectZero` when set, then a default frame size will be used.

\* Set this value to `nil` to remove the button.

\*/

@property (weak, nonatomic) UIButton \*leftBarButtonItem;

/\*\*

\* Specifies the width of the leftBarButtonItem.

\*

\* @discussion This property modifies the width of the leftBarButtonContainerView.

\*/

@property (assign, nonatomic) CGFloat leftBarButtonItemWidth;

/\*\*

\* Specifies the amount of spacing between the content view and the leading edge of leftBarButtonItem.

\*

\* @discussion The default value is `8.0f`.

\*/

@property (assign, nonatomic) CGFloat leftContentPadding;

/\*\*

\* The container view for the leftBarButtonItem.

\*

\* @discussion

\* You may use this property to add additional button items to the left side of the toolbar content view.

\* However, you will be completely responsible for responding to all touch events for these buttons

\* in your `JSQMessagesViewController` subclass.

\*/

@property (weak, nonatomic, readonly) UIView \*leftBarButtonContainerView;

/\*\*

\* A custom button item displayed on the right of the toolbar content view.

\*

\* @discussion The frame height of this button is ignored. When you set this property, the button

\* is fitted within a pre-defined default content view, the rightBarButtonContainerView,

\* whose height is determined by the height of the toolbar. However, the width of this button

\* will be preserved. You may specify a new width using `rightBarButtonItemWidth`.

\* If the frame of this button is equal to `CGRectZero` when set, then a default frame size will be used.

\* Set this value to `nil` to remove the button.

\*/

@property (weak, nonatomic) UIButton \*rightBarButtonItem;

/\*\*

\* Specifies the width of the rightBarButtonItem.

\*

\* @discussion This property modifies the width of the rightBarButtonContainerView.

\*/

@property (assign, nonatomic) CGFloat rightBarButtonItemWidth;

/\*\*

\* Specifies the amount of spacing between the content view and the trailing edge of rightBarButtonItem.

\*

\* @discussion The default value is `8.0f`.

\*/

@property (assign, nonatomic) CGFloat rightContentPadding;

/\*\*

\* The container view for the rightBarButtonItem.

\*

\* @discussion

\* You may use this property to add additional button items to the right side of the toolbar content view.

\* However, you will be completely responsible for responding to all touch events for these buttons

\* in your `JSQMessagesViewController` subclass.

\*/

@property (weak, nonatomic, readonly) UIView \*rightBarButtonContainerView;

#pragma mark - Class methods

/\*\*

\* Returns the `UINib` object initialized for a `JSQMessagesToolbarContentView`.

\*

\* @return The initialized `UINib` object or `nil` if there were errors during

\* initialization or the nib file could not be located.

\*/

+ (UINib \*)nib;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

/\*\*

\* A constant defining the default height of a `JSQMessagesTypingIndicatorFooterView`.

\*/

FOUNDATION\_EXPORT const CGFloat kJSQMessagesTypingIndicatorFooterViewHeight;

/\*\*

\* The `JSQMessagesTypingIndicatorFooterView` class implements a reusable view that can be placed

\* at the bottom of a `JSQMessagesCollectionView`. This view represents a typing indicator

\* for incoming messages.

\*/

@interface JSQMessagesTypingIndicatorFooterView : UICollectionReusableView

#pragma mark - Class methods

/\*\*

\* Returns the `UINib` object initialized for the collection reusable view.

\*

\* @return The initialized `UINib` object or `nil` if there were errors during

\* initialization or the nib file could not be located.

\*/

+ (UINib \*)nib;

/\*\*

\* Returns the default string used to identify the reusable footer view.

\*

\* @return The string used to identify the reusable footer view.

\*/

+ (NSString \*)footerReuseIdentifier;

#pragma mark - Typing indicator

/\*\*

\* Configures the receiver with the specified attributes for the given collection view.

\* Call this method after dequeuing the footer view.

\*

\* @param ellipsisColor The color of the typing indicator ellipsis. This value must not be `nil`.

\* @param messageBubbleColor The color of the typing indicator message bubble. This value must not be `nil`.

\* @param shouldDisplayOnLeft Specifies whether the typing indicator displays on the left or right side of the collection view when displayed.

\* @param collectionView The collection view in which the footer view will appear. This value must not be `nil`.

\*/

- (void)configureWithEllipsisColor:(UIColor \*)ellipsisColor

messageBubbleColor:(UIColor \*)messageBubbleColor

shouldDisplayOnLeft:(BOOL)shouldDisplayOnLeft

forCollectionView:(UICollectionView \*)collectionView;

@end

//

// MasterViewController.h

// AIChatBots

//

// Created by yangboz on 2016/12/26.

// Copyright © 2016年 \_\_\_SMARTKIT.INFO\_\_\_. All rights reserved.

//

#import <UIKit/UIKit.h>

#import <iAd/iAd.h>

#import "DataModel.h"

@class DetailViewController;

@interface MasterViewController : UITableViewController

{

NSMutableArray \*\_chatbots;

}

@property (strong, nonatomic) DetailViewController \*detailViewController;

@property (nonatomic, retain) NSMutableArray \*chatbots;

@property (nonatomic, retain) NSMutableArray \*listOfRatings;

@property (nonatomic, retain) NSMutableArray \*groupedChatbots;

@end

//

// MessageVoModel.h

// AIChatBots

//

// Created by yangboz on 2016/12/27.

// Copyright © 2016年 \_\_\_SMARTKIT.INFO\_\_\_. All rights reserved.

//

#import <UIKit/UIKit.h>

#import "JSONModel.h"

@interface MessageVoModel:JSONModel

@property(nonatomic) NSString \*message;

@property(nonatomic) NSNumber \*chatBotID;

@property(nonatomic) NSString \*chatBotName;

@property(nonatomic) NSString \*emotion;

@end

//

// MessageVoSimple.h

// AIChatBots

//

// Created by yangboz on 11/05/2017.

// Copyright © 2017 \_\_\_SMARTKIT.INFO\_\_\_. All rights reserved.

//

#import <Foundation/Foundation.h>

@interface MessageVoSimple : NSObject

@property(nonatomic) NSString \*message;

@property(nonatomic) NSNumber \*chatBotID;

@property(nonatomic) NSString \*emotion;

@property(nonatomic) NSString \*chatBotName;

@end

//

// Created by Jesse Squires

// http://www.jessesquires.com

//

//

// Documentation

// http://cocoadocs.org/docsets/JSQMessagesViewController

//

//

// GitHub

// https://github.com/jessesquires/JSQMessagesViewController

//

//

// License

// Copyright (c) 2014 Jesse Squires

// Released under an MIT license: http://opensource.org/licenses/MIT

//

#import <Foundation/Foundation.h>

@interface NSUserDefaults (DemoSettings)

+ (void)saveExtraMessagesSetting:(BOOL)value;

+ (BOOL)extraMessagesSetting;

+ (void)saveLongMessageSetting:(BOOL)value;

+ (BOOL)longMessageSetting;

+ (void)saveEmptyMessagesSetting:(BOOL)value;

+ (BOOL)emptyMessagesSetting;

+ (void)saveSpringinessSetting:(BOOL)value;

+ (BOOL)springinessSetting;

+ (void)saveOutgoingAvatarSetting:(BOOL)value;

+ (BOOL)outgoingAvatarSetting;

+ (void)saveIncomingAvatarSetting:(BOOL)value;

+ (BOOL)incomingAvatarSetting;

+ (void)saveAccessoryButtonForMediaMessages:(BOOL)value;

+ (BOOL)accessoryButtonForMediaMessages;

@end

//

// ResponseVoModel.h

// AIChatBots

//

// Created by yangboz on 2016/12/28.

// Copyright © 2016年 \_\_\_SMARTKIT.INFO\_\_\_. All rights reserved.

//

#import <Foundation/Foundation.h>

#import "JSONModel.h"

#import "MessageVoModel.h"

@protocol MessageVoModel

@end

@interface ResponseVoModel : JSONModel

@property(nonatomic) NSNumber \*success;

//@property(nonatomic,retain) NSString \*errorType;

@property(nonatomic) NSString \*errorMessage;

@property(nonatomic) MessageVoModel \*message;

@end

//

// StringUtil.h

// ChatBotsMessager

//

// Created by yangboz on 14-5-18.

// Copyright (c) 2014年 GODPAPER. All rights reserved.

//

#import <Foundation/Foundation.h>

#import <CommonCrypto/CommonDigest.h>

#import <CommonCrypto/CommonHMAC.h>

@interface StringUtil : NSObject

{}

//@see http://stackoverflow.com/questions/476455/is-there-a-library-for-iphone-to-work-with-hmac-sha-1-encoding

+ (NSString\*) HMACWithSecret:(NSString\*)secret andData:(NSString \*)data;

//

+ (NSString \*)HMACSHA1withKey:(NSString \*)secret forString:(NSString \*)data;

//

+(NSString \*)HMACSHA256withKey:(NSString \*)plaintext withKey:(NSString \*)key;

@end

//

// UIImageUtils.h

// DeepDetectChatBots

//

// Created by yangboz on 30/01/2017.

// Copyright © 2017 \_\_\_SMARTKIT.INFO\_\_\_. All rights reserved.

//

#import <Foundation/Foundation.h>

#import <UIKit/UIKit.h>

@interface UIImageUtils : NSObject

+ (UIImage \*)imageWithImage:(UIImage \*)image scaledToSize:(CGSize)newSize;

+ (UIImage\*) grayishImage: (UIImage\*) inputImage;

@end

//

// VC\_Segue\_AppInfo.h

// ChatBotsMessager

//

// Created by yangboz on 13-11-5.

// Copyright (c) 2013年 GODPAPER. All rights reserved.

//

#import <UIKit/UIKit.h>

@interface VC\_Segue\_AppInfo : UITableViewController

{}

- (IBAction)dissmiss:(id)sender;

@property(nonatomic,strong) IBOutlet UITextView \*appInfoTxtView;

@end

//

// VC\_Segue\_ChatBotInfo.h

// ChatBotsMessager

//

// Created by yangboz on 13-11-5.

// Copyright (c) 2013年 GODPAPER. All rights reserved.

//

#import <UIKit/UIKit.h>

#import "DataModel.h"

#import "ChatBotVoModel.h"

@interface VC\_Segue\_ChatBotInfo : UITableViewController

{}

- (IBAction)dissmiss:(id)sender;

@end