Emojier SDK for iOS Interface Documentation

1.Dependency

1) AFNetWorking

2) Colours

3) FMDB

4) MJExtension

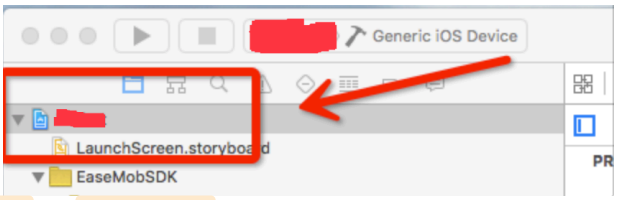
5) RegexKitLite

6) SDWebImage

You can get all them from github.

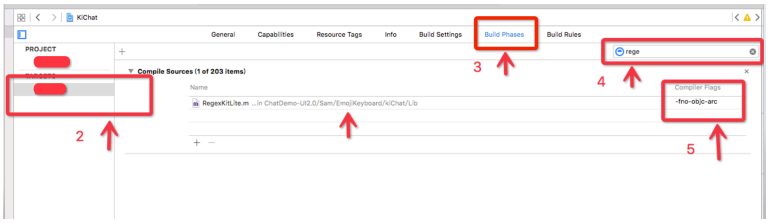
2.Add -fno-objc-arc for RegexKitLite in Xcode project

1) `Your project`



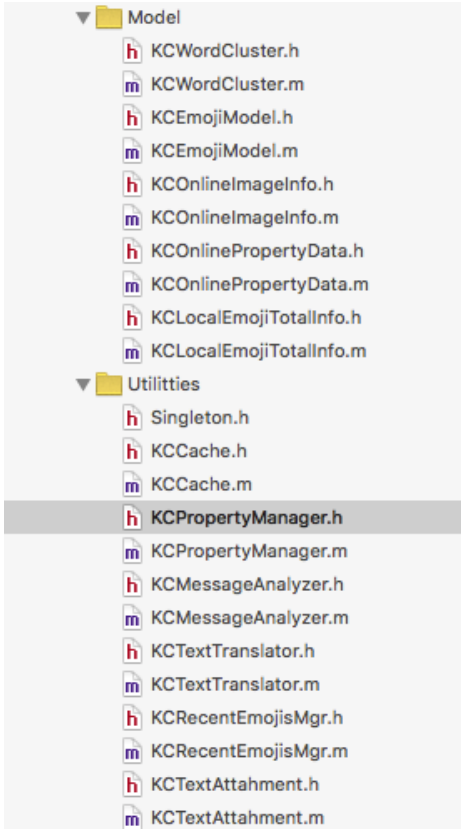
2) `TARGETS` ---- `Your Target` ---- `Build Phases`

3)Input `RegexKitLite` in the search with the place holder "Filter", then you will find `RegexKitLite.m` below. Now input `-fno-objc-arc` below the `Compiler Flags`.



3.The core code

You need `Model` and `Utilities` for your project as below:



4.How to use those Classes

4.1 APPKEY & User Key

1) APPKEY: required. If it lost, you can do nothing. You must change the APPKEY in the `KCPropertyManager.h` first before you do anything.You just need to replace the `testappkey1` string with your APPKEY(How to get app key, please refer to emojier support).

2) User Key: It is necessary when user want to customize their own private emojis on our website to replace the offical one, if he did, he will get an unique private `User Key`,then you need to provide somewhere for the user to input and store this key.Then you just need to call set [KCPropertyManager sharePropertyMgr] user key(How to get app key, please refer to emojier support).

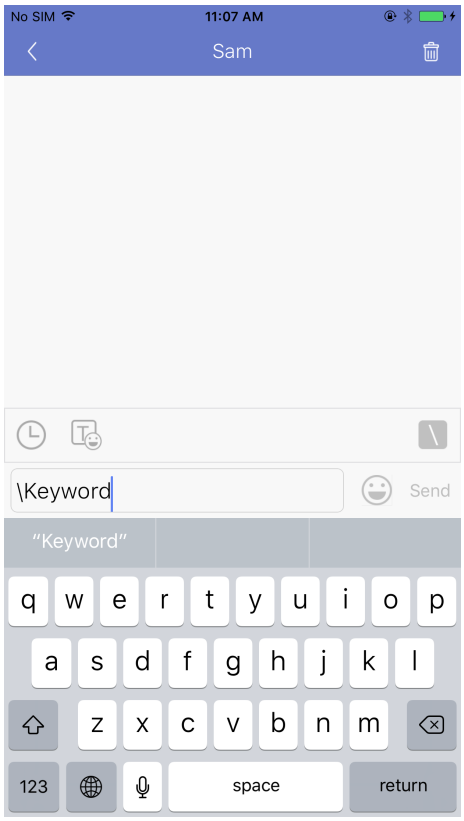
```
[[KCPropertyManager sharePropertyMgr] setUserKey:userKey];
```

You don't worry about the key is nil or it is right or not. If the key is wrong or nil, you just can't get customized emojis.

```
// Your APP_KEY
#define APP_KEY @"testappkey1"
```

4.2 Convert single word to emoji

This function is useful when you only one or more word are converted to emoji instead of whole sentence are converted.



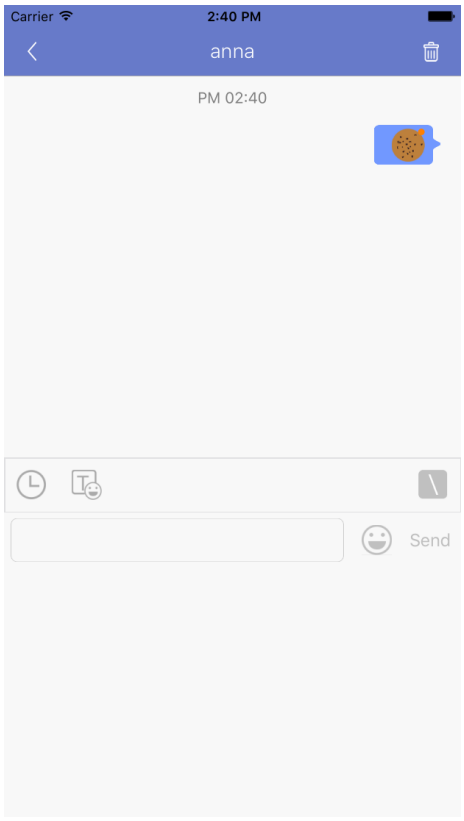

```
 [[KCPropertyManager sharePropertyMgr] requestPropertyArrayWithKeyword:keyword emojiModelBlk:^(NSMutableArray
*emojiModelArray) {
 // Here you can get the emojis, then you can put them/it on the tool bar.
    }];
```

The `emojiModelArray` is array of `KCEmojiModel`, the picture below shows properties of `KCEmojiModel`.

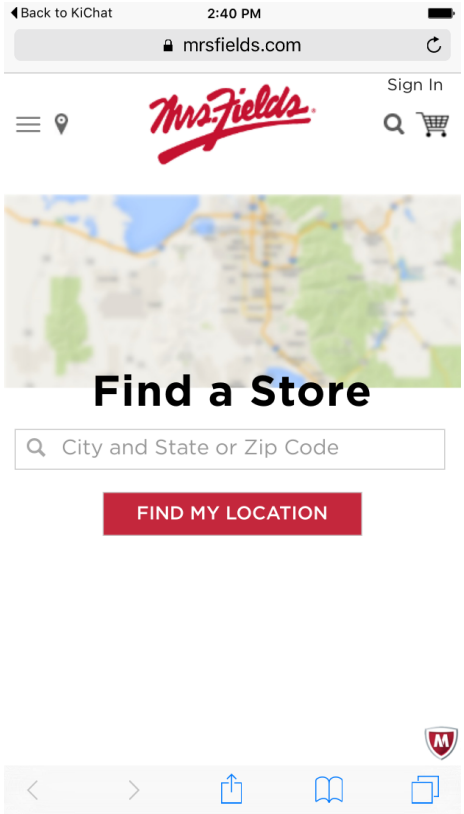
```
@property (nonatomic, assign) SMEmojiType emojiType;
@property (nonatomic, strong) NSString *emojiImageName;
@property (nonatomic, strong) NSString  *property_id;
@property (nonatomic, strong) NSString *link;
@property (nonatomic, strong) NSString *keyword;
@property (nonatomic, assign) SMComeFrom comeFrom;
```

`emojiType` where the emoji image come from: online or local
`emojiImageName` is the picture's name
`property_id` the online emojiModel has a unique propertyID, this property is nil to the local ones
`keyword` 🐶's keyword is `dog`,sometimes it is useful
Usually, you needn't concern with the above four properties, `link` and `comeFrom` maybe are important to you.

`link` it's also the online emojiModel's property. For example, Sam send a emoji `cookie` to James, James want to know where to buy it, he can just tap the emoji of `cookie` he has received, then he can open the `link`, and see the sale information on a website. These things were finished by `KCMessageAnalyzer` class as mentioned below in `Part 5 Analyze the dialog list`.Which is important, emojis that have a link, in another word, emojis that can be clicked have an orange dot on the top-right corner. Just as the picture shows:

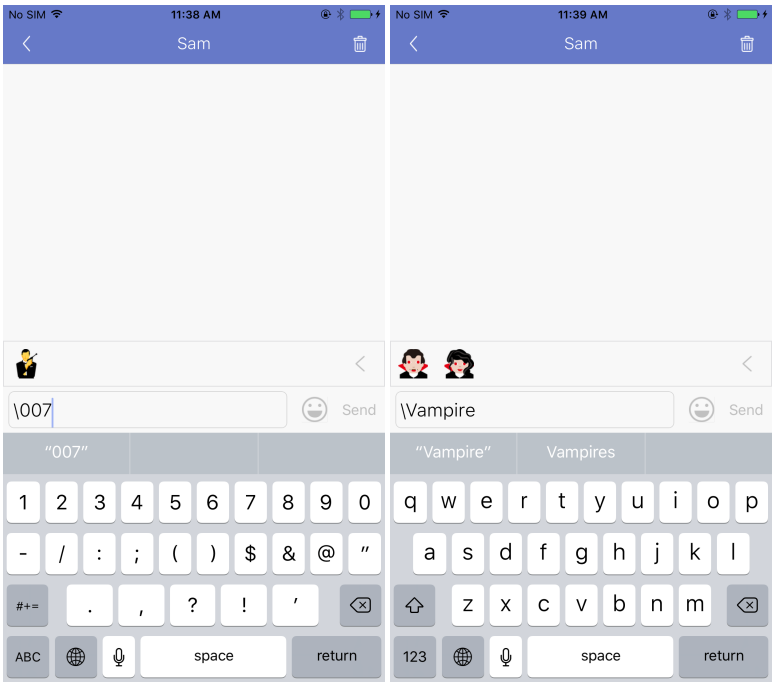


If user tap the `Cookie` emoji, the website will be opened:



`comeFrom` maybe is an important property.When user tap an emoji, the emoji can be from `Recent Emoijs`, or `\Keyword` result, or sentence conversion result. If it comes from `Recent Emoijs`, you just need to insert the attributedString at the end of the

textView, and there's no need to delete anything. If it is `\Keyword` or sentence conversion result, you should delete the keyword, and then insert the attributed string at the index of the keyword. Now we regard `\Keyword` and sentence conversion result both as `SMComeFromTranslate`. You don't need to specify the type of any emoji anywhere, because the SDK has specify `SMComeFromRecent` `comeFrom` for `Recent Emoijs` when store emojis to sandbox, and specify `SMComeFromTranslate` `comeFrom` for `\Keyword` result and translate result when translate.You just need to use this property to handle your text in the textView.



Here we got a problem: how to convert an emojiModel to an attributedString and insert it in to the textView? You have two choices.

1.If you'd like SDK to handle your all text in your textView, you just call the method below, then the SDK will help to handle the str.

```
+ (void)translateWhenUserClickedEmoji:(KCEmojiModel *)recent withSourceAttrubutedStr:(NSMutableAttributedString *)str content:(NSString *)content
```

@param `emoji` emoji that user select
@param `str` (NSMutableAttributedString *)textView.attributedText
@param `content` textView.text

2.If you do not want the SDK to know your total text, you can call the method below to convert the emojiModel to an attributedString, but you should handle the something such delete the keyword yourself.

```
+(NSMutableAttributedString *)textAttachmentStringWithEmojiModel:(KCEmojiModel *)emojiModel
```

4.3 Recent Emoji

If you want to your app has recent emoji feature. First, you should track the instance of KCEmojiModel for each emoji, then convert it to a dict, so that the `KCRecentEmojisMgr` can store it to the sandbox.

1.Convert to dict

```
NSDictionary * dict = [KCEmojiModel emojiModelWithEmoji:recent type:recent.emojiType];
```

2.Append to KCRecentEmojisMgr

```
    [[KCRecentEmojisMgr shareRecentEmojiMgr] updateRecentEmojisWithResentEmoji:recent withDict:dict];
```

3.Get all recent emoji
You can call `- (NSArray *)recentEmojiModelArray;` to get all you had appended.

```
NSArray *recentArray = [[KCRecentEmojisMgr shareRecentEmojiMgr] recentEmojiModelArray];
```

When you get the recentArray, you can decide how to handle it. Ex: display it in tool bar. The image path for each recent emoji defined by `ImageCachePath`, you can find it in `KCPropertyManager.h`

```
    #define ImageCachePath [[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES)
lastObject]stringByAppendingPathComponent:@"emojiImage"]
```

4.4 Convert sentence to a series of emojis

If you want to convert one more sentences to a series of emojis in one time, you need `KCTextTranslator`. There are two main methods:
`translateSourcAttributedStr:normalStr:withHighlightColor:imageSize:translateOneTime:toDestionAttributedStr:`
`resotreNormalStringWithAttributedString:andNormalText:`
1)
`translateSourcAttributedStr:normalStr:withHighlightColor:imageSize:translateOneTime:toDestionAttributedStr:`
This method is used to convert sentence to display format which can be present by os system. It doesn't matter if part of sentence had been converted to emojis. when conversion done, your callback will be called with the final Attributed String and if the sentence contains words that have more than one emojis. For detail parameters and prototype, please refer to below:

@param `sourceAttrs` textView.attirebutedText (can't be nil)

@param `highlightColor` If there are keywords have more than one emojis, the keyword will be highlighted, so you need set what color you want.

@param `imageSize` When translate a keyword into an image(Actually it is an attributedString with an `KCAttachment`), you need to set the image size. `Default is (30,30).`

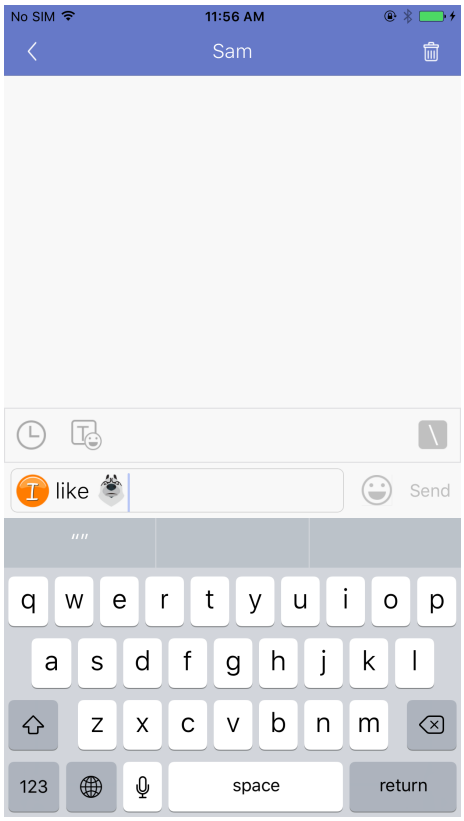
@param `normalStr` textView.text (can't be nil)

@param `translateOneTime` As described above, some words may have have more than one emojis, if `translateOneTime` is `NO`, the keyword will be replaced by a the first emoji, otherwise the keyword will be highlighted so that user can tap the keyword and chose one emoji. there is no need to specify `highlightColor` When `translateOneTime` is `NO`.

@param `result` final result

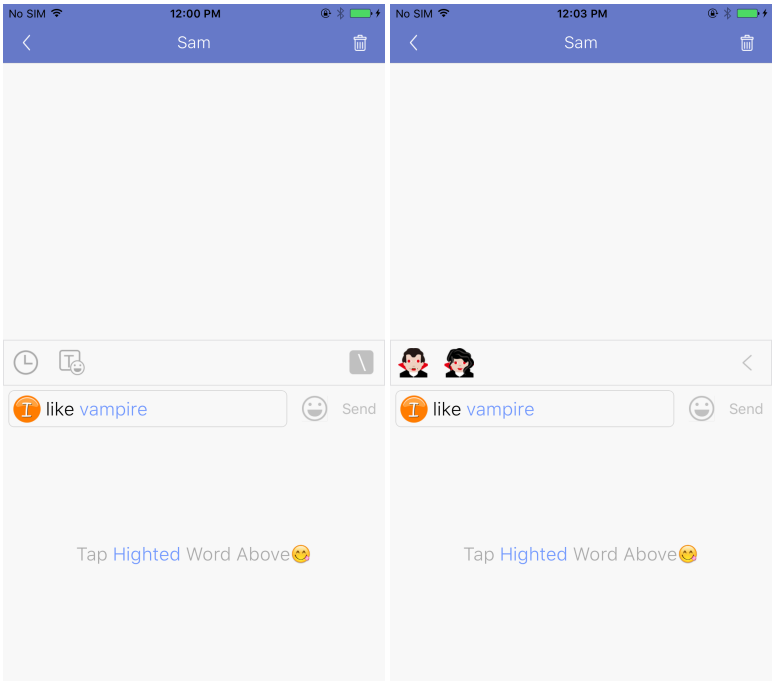
```
    + (void)translateSourcAttributedStr:(NSAttributedString *)sourceAttrs normalStr:(NSString *)normalStr
withHighlightColor:(UIColor *)highlightColor imageSize:(CGSize )imageSize translateOneTime:
(BOOL)translateOneTime toDestionAttributedStr:(void (^)(NSAttributedString *destiAttrs,BOOL
hasMutiEmojis))result;
```

For example, `I like dog` ---> convert, you will get the result:The attributed string in the textView is `destiAttrs` and `hasMutiEmojis` is `NO`, it means there are no keywords which have more than one emojis.



`I like vampire` ---> convert, you will get the result:The attributed string in the textView is `destiAttrs` and `hasMutiEmojis` is `YES`, There are two different emojis for `vampire`, so it is highlighted and wait user to select one. When user touch the highlighted area, the emojis will show on the tool bar.

So when user cliked somewhere, if the area is highlighted, you should find out the world in the rect, and then just do what you do after user input `\keyword`.



2) `resotreNormalStringWithAttributedString:andNormalText:`

Before you send your message to another user, ex: click `send`, you should call this method to encode attributedString to normalString.

When click `send` you should restore the attributedString to normalString.

@param `attributedStr` textView.attirebutedText

@param `normalText` textView.text

@return finalNormalString

```
+ (NSString *)resotreNormalStringWithAttributedString:(NSAttributedString *)attributedStr andNormalText:
(NSString *)normalText;
```

4.5 Decode normalText string

You should call this method in every tableView cell when analyze the dialog list. When all emojis are downloaded,KCPropertyManager will post a notification `DownloadImageByPropertyIDDoneNotification`, so you can reload the data of the dialog list.

@param `contentStr` cell.label.text
@param `messageID` unique messageID to do cell size cache
@param `sourceAttS` cell.label.attributedText if the size is already cached, there's no need to analyze the text again
@param `fontColor` if there are words have more than one emojis, it will be highlighted with the color
@return the final attributedStr which already been convert to attributedStr

```
+(NSMutableAttributedString *)attributedStringWithContentString:(NSString *)contentStr withMessageID:(NSString
*)messageID withAttributedStr:(NSAttributedString *)sourceAttS withFontColor:(UIColor *)fontColor;
```

If you need to calculate the tableViewCell's height with the textLabel/textView size, the below method is helpful.

@param `content` textLabel.text
@param `messageID` unique messageID
@param `sourceAttS` textLabel.attributedText
`messageID` and `sourceAttS` is used to get cache data

@return textLabelSize

```
+(CGSize)sizeWithContent:(NSString *)content withMessageID:(NSString *)messageID withAttributedStr:
(NSAttributedString *)sourceAttS;
```

46 Delete images if they taking up too much space

There are two methods in `KCPropertyManager.h` you can call.
1) `- (void)clearImageCache;`
This method will clear all the images that have been cached in the sandbox.
2) `- (void)clearImagesIfMoreThanSize:(long long)maxSize;`
This method will clean image if all image size > maxSize(Bytes). Images will be sorted by last use time, the earlier used will be deleted until all images size <= maxSize * 0.5 .