

第5回

「アプリ作りと永久回廊(1)と」

toru.inoue@kissaki.tv

Agenda

- 1.プログラミング言語 座学 ファイナル
- 2.セーブ・セーバー・セーブスト
- 3.Let's Rock (プログラムしましょう)
- 4.「GitHub、君に決めた！」

Drive I / 35

I.プログラミング言語 座学 ファイナル

Drive2/35

1-0-0.ヘッダファイル

別に山道をコップの水気にしながら走る訳じゃない

.mはクラスファイルだった。.hは、じゃあなんだ？

hは、header の頭文字のHです。

ヘッダーファイル、単にヘッダとか言います。

ヘッダー 【header】

文書の用紙の上部に定型として印刷される、タイトルや日付などの文字列。



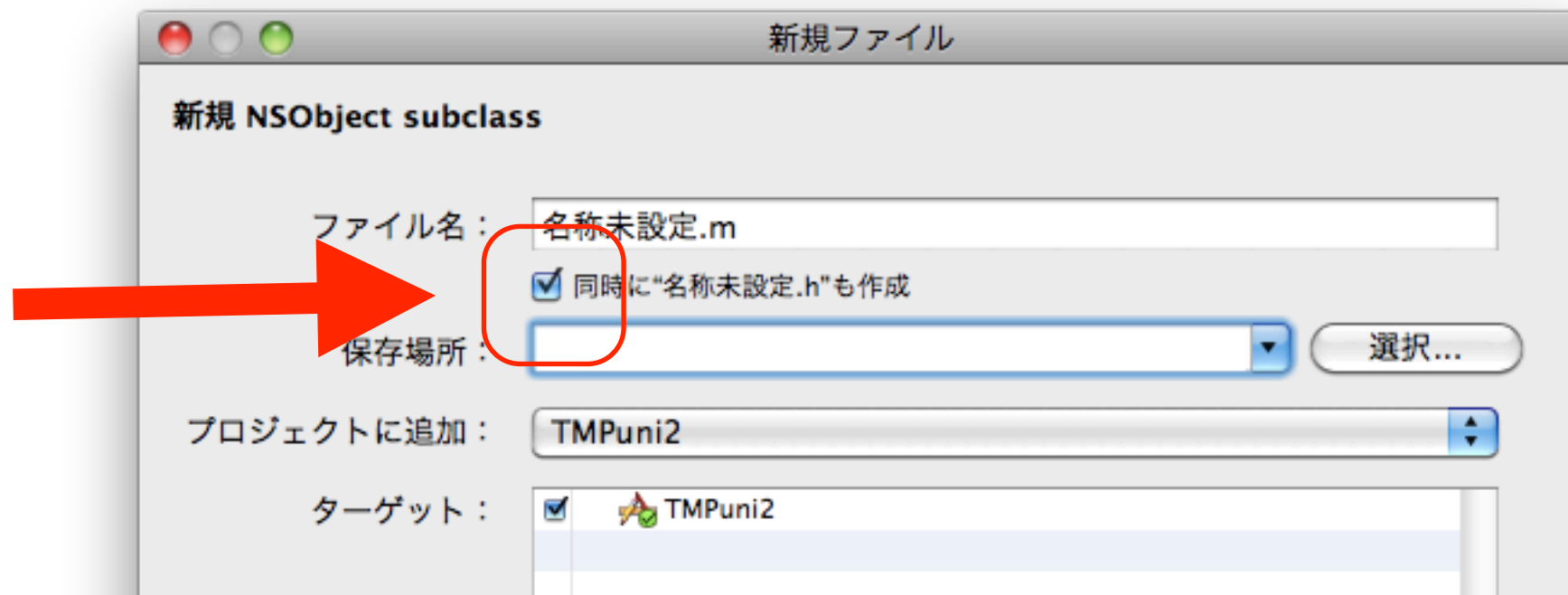
Drive3/35

1-0-0.ヘッダファイル

あなたのためだから、は大概ただの嫌がらせ

いままで全く伝えずに続けてきたのは、
.hファイルがいろいろな内容を含んだ、実に包括的な代物だからです。

.hは、.mファイルを作るときに、Xcodeががんばって作ってくれます。



試しに、どこでもいいのでAAAという名前のファイルを、
上記のチェックを入れて作ってみましょう。

Drive4/35

1-0-0.ヘッダファイル

完全に一致

Xcodeさんが巧いことやってくれと、ファイル名は一致してて、
拡張子だけが違う.mと.hファイルが出来上がります。

この2つのファイル、名前が完全に一致

AAA.m ←→ AAA.h

AAA.mファイル
[.m]

AAA.hファイル
[.h]

(完全に一致)



Drive5/35

1-0-2.ヘッダファイルの役割

試しにAAAという名前のファイルをつくってみた。

よく見る.hファイル。こんな感じだよねー。

```
#import <UIKit/UIKit.h>

@interface AAA : NSObject {
}

@end
```

```
#import <UIKit/UIKit.h>
と、
@interface
と、
@end
```

っつー要素が、あります。

Drive6/35

1-0-2.ヘッダファイルの役割

で、ついでにクラスファイル(.m)も見ると、

```
#import "AAA.h"
```

とか、何やら最初からしゃあしゃあと書いてあります。

```
#import "AAA.h"
```

```
@implementation AAA
```

```
.....
```

```
@end
```

```
#import "AAA.h"
```

.hに続いて、こっちにもimport、でも内容が "AAA.h"

あと、

```
@implementation
```

```
@end
```

とか。 こいつら、順に見て行きましょうか。

一度.h(ヘッダー)の説明から離れて、その内容になっているものを見ます。

#importから。

Drive7/35

1-0-3.インポート

#import 余裕のある生活.h → no such resource エラー orz

#import

散々いままでもでてきましたが、とうとうこいつの説明をします。
プログラムの中で、

他のクラスの事を書く際、
そのクラスの情報、自分のクラスに持ち込む(import)必要があります。

そこで、

#import "クラス名.なんやら" ってやると、夢が叶います。

```
#import "XXX.なんやら"  
BBB * bbb = ...
```

ファイル
[.なんやら]

AAA.なんやら

ファイル
[.なんやら]

BBB.なんやら

Drive8/35

1-0-4.インポート品って大体雑に扱われるよね

```
#import "クラス名.h"
```

以外にも、

```
#import <UIKit/UIKit.h>
```

とか、””でなく<>を使う場合もあります。

意味はもちろん異なります。→ちょっとあとでやります。

ライブラリ、frameworkの項にて。

じゃ、次は@なんちゃらとかに行きますかね。

Drive9/35

1-0-5.@interfaceと@implementation

今まで恐ろしいくらいシカトしててヒヤヒヤ
やっそこ説明。

.hに、@interface @end

```
@interface AAA : NSObject {  
}  
@end
```

.mに、@implementation @end

```
#import "AAA.h"  
@implementation AAA  
.....  
@end
```

で、こいつは、.hと.mにそれぞれ書く事が決まっている、訳ではありません。

Drive 10/35

1-0-6.ここから～ と、～ここまで

Objective-Cでは、@~,@end の間に、いろいろな内容を書き込む、という記述形式になってます。AAAというクラスを作って、例としてやりましょ。

@interfaceには、このAAAというクラスで使うオブジェクトの内容を{ }の中に書いて、

このAAAというクラスで使うメソッドの情報(interface)を@endまでの間に書く事ができます。

```
@interface AAA : NSObject { //この間にオブジェクト書ける
    BBB * b;
    int m_myInt;
}
//この間にメソッドの情報を書けます
- (void) justDoIt;
@end
```

Drive I I / 35

1-0-6.ここから～ と、～ここまで

`@implementation`には、このAAAというクラスで使う
メソッドの実装(implementation)や、
オブジェクトを扱って実際に何かする、プログラムの内容を、
`@end`までの間に書く事ができます。

```
#import "AAA.h"
@implementation AAA
//この間にメソッドが書ける

- (void) justDoIt {
    bbb = [[BBB alloc] init];
    m_myInt = 100;
}

@end
```

`#import "AAA.h"`がさりげなく有る事に注目。意味はすぐに判ります。

Drive | 2/35

1-1-0. 「実は」、という始まり方は、「実は」、よくない
せんぱい、ちょっと冷静に自分の言ってる事考えてみてくださいorz

.hに@interface

.mに@implementation

とか有るのですが、

@interfaceと@implementationとを

一つの.mファイルに書いても、きちんと動きます。

Xcodeの「同時に~~.hも作成」チェックを外して

ファイルを作ってやってみると、まあ、いろいろやれば動きます。

次のページにソースコード書いてみました。

Drive I 3/35

1-1-1.

これが、まあ、その、アレだ

AAA.m

```
//まずinterface
@interface AAA : NSObject {

}

- (void) doTest;

@end

//ここからimplementation
@implementation AAA

- (void) doTest {
}

@end
```

#import "AAA.h"が無くても、動きます。

Drive I 4/35

1-2-0.一つのファイルに書いていいのか？ 大丈夫だ、問題ない。
おまけに`#import "AAA.h"`も消えて、ラッキー！

- ・じゃあなんでそうしないのか→



簡単にいうと、取り回しが悪いから。具体例はあるがまあ、挙げると

①.hがあると、一つの.mが太らない（ぐっちゃんにならないで済む）

②オブジェクト、インスタンスの名前で困らなくなる(名前が重複してもok)

とか。

Drive | 5/35

1-2-0.ヘッダまとめ

.hファイルは、無くてもプログラムを書く事が出来ます。
ただし、その代償としてのデメリットが多々あり、
それを避けるために、.mファイルと.hファイルが別々に用意され、
.mファイルには@implementationが、
.hファイルには@interfaceが、それぞれ

「分けて書けるように」なっています。

「分けて書いた方が取り回しがよく、拡張できるので.hがある」、
「分けて書く為に@implementationと@interfaceがある」
という順番に考えると判りやすいと思います。

つまり、

拡張性などの利益追求の為に、

.h(ヘッダ)ファイルと.m(クラス)ファイルに別れてるんですわ。

Drive | 6/35

1-2-0.循環参照

遷移を作ってて、
作ったクラスの.hにAppDelegate.hって書いたら動かなくなりました
って人が居たら、解説します。

Drive | 7/35

1-2-0.その他の値、書き方

#define

C言語から由来のもので、

#define

なんてのがあります。なんかしら、使い手が好きな内容をdefine(定義)できるものです。

#define PARAM_GRAVITY (0.98)

こんな風を書いておくと、

プログラムがビルドされる前に、**PARAM_GRAVITY**って書いてあるところが、自動的に**(0.98)**に書き変わります。 うわー便利。

ぶっちゃけ、一見ただの置換です。

よって、式とか、文字も書けます。

Drive | 8/35

1-2-1. 頻繁に調整したい値などがあるとき、defineで書いておくと、ソースコードの中の数値とかを直に書き換えなくて変更できる。

名前 と、 値 の間は、タブかスペースがおすすめです。

()をつけるのは、まあ、やってみると判るんですが、数値系の場合、付けないとバグの元になりやすいからです。

無しでdefineやってる方はうちのプログラムの師匠から情弱認定されます。

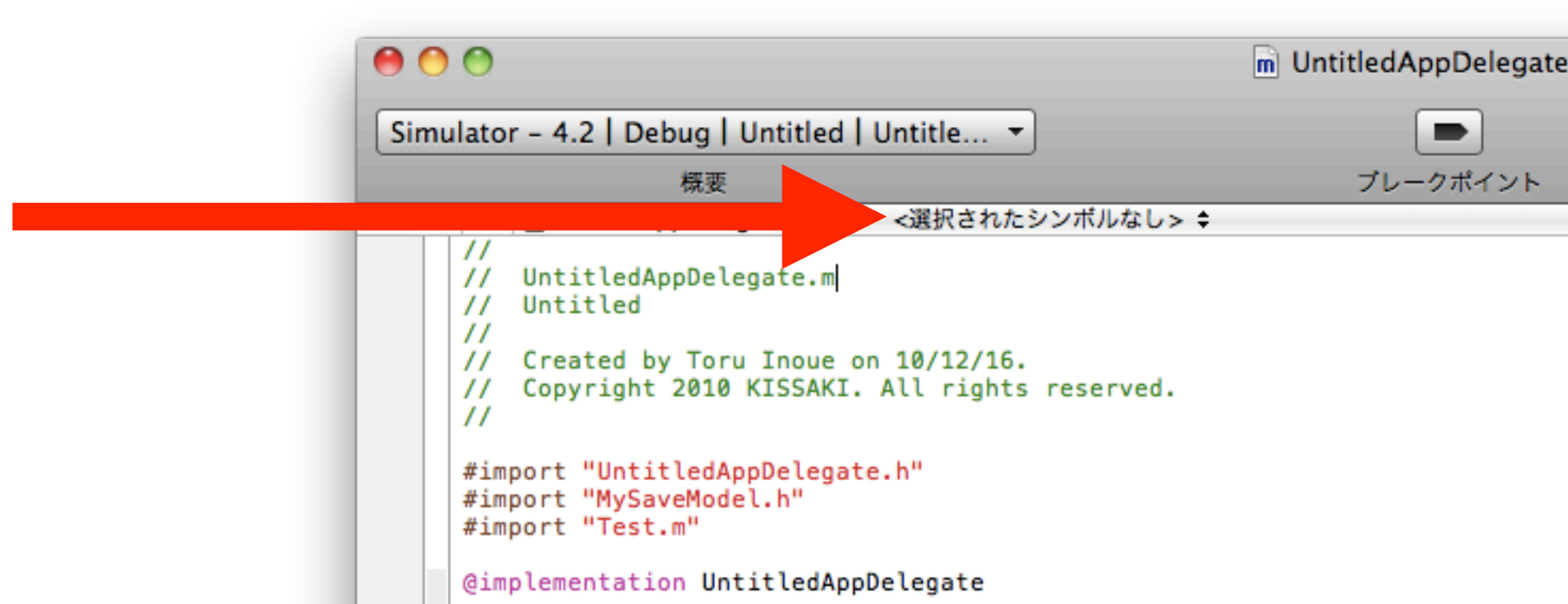
Drive | 9/35

1-2-2. #pragma

素人が使うな。玄人も意図限定した上で、でもやっぱ使うな。

ミスリードの元になる。

ソースコードとの最新一致性が保てない。使うな。以上。



選択されたシンボルなし、とかから選ぶとき、表示されます。

他にもいろんな、アレな、機能が満載ですが、

コードの読みやすさを下げるだけです。

使うな。

Drive20/35

1-2-3. コメント

、、、まさかこんな所を書く事になるとは、 、 こんどから気をつけゆ。

// 半角スラッシュを二つ並べて書くと、その行のスラッシュから後ろは、
プログラムに関わらない
コメントになります。

/* 半角スラッシュ米
と

*/ 米半角スラッシュ
で囲んだ範囲も、コメントになります。

プログラム中で気になった事や、あとで自分が見たときに「あーここ説明あったほうがいいなー」

とか思った部分について、書いておくと、いいです。

Xcodeだと、⌘ + / でコメントアウト出来ます。もっかいやると解除。

Drive21/35

1-3-0. コメントについてのコメント

＊但し

コメントとプログラムは、関係ありません。

コメントがなくてもプログラムは動きます。

プログラムを変えてもコメントは自動的に変わりません。

プログラムを書き換えて、内容が異なる物になった時、
コメントを合わせて書き直さなかったりすると、

コメントに書いてあるものとプログラムとは、別の内容になってしまいます。
コメントで余計な先入観を与えないように、気をつけてね！

Drive22/35

1-4-0.ライブラリ、framework

```
#import "クラス名.なににやら"
```

以外にも、

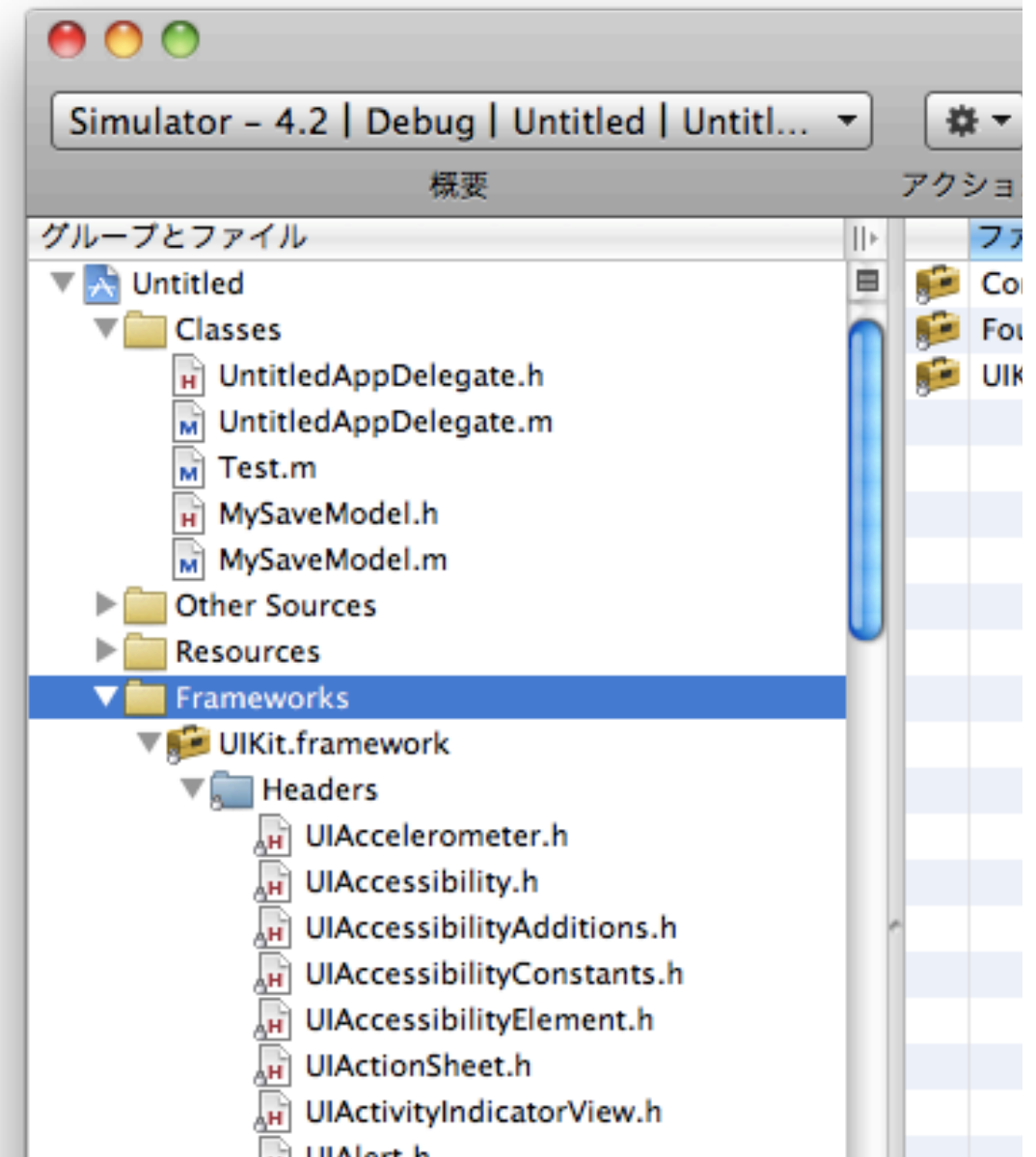
```
#import <UIKit/UIKit.h>
```

とか、""でなく<>を使う場合、

Appleが用意した、Frameworkという
.hのカタマリをimportするときは、
<>を使います。

簡単に言うと、
自分で作ったもの、.mが手元に有るもの
→""でimport

それ以外にだれかが作ったもの
→<>でimport
とか。あくまで簡単に言うと。



FrameworkはPWでその内容を確認出来ます。

Drive I / 35

2.セーブ、セーバー、セーベスト

Drive24/35

1-4-1.保存ライブラリ作り、セーブとロード

ゆとりのあるセーブ仕様として、NSUserDefaultsを使います。

NSUserDefaultsを使ってみよう(CoreDataはまだまだ無理だろJK)

アプリケーションを使っていると、

使ったときの内容を保存したくなったりとか、します。

で、実際その部分は、全員必須なので、作ってみましょう。

Drive25/35

1-4-2.書いてみる

NSUserDefaults というクラスを使うと、望みの事が出来ます。

```
NSUserDefaults * userDef = [[NSUserDefaults alloc] init];  
[userDef setValue:@"あたたた" forKey:@"俺の名前を言ってみろ"]; //書き込み  
NSString * str = [userDef valueForKey:@"俺の名前を言ってみろ"]; //読み出し  
NSLog(@"str_%@",str); //確認
```

→キーというなんかヒントになる物を用意して、
そのヒントから得られるものに値をセットします。

この場合、キー：@"俺の名前を言ってみろ" に、値：@"あたたた"をセット、
同時にその組を保存してます。

んーでも、本当に保存されてるのかな。 書いてあるから出てるだけじゃね？

Drive26/35

1-4-3.今度は書き込まずに、ただ読み出ししてみる
書き込んでる部分をコメントアウトしても取り出せれば、保存okなはず

```
NSUserDefaults * userDef = [[NSUserDefaults alloc] init];  
// [userDef setValue:@"あたた" forKey:@"俺の名前を言ってみる"]; //書き込み  
NSString * str = [userDef valueForKey:@"俺の名前を言ってみる"]; //読み出し  
NSLog(@"str_%@",str); //確認
```

→キーに対応したオブジェクトがuserDefに保存されているので、
以前保存しておいたパラメータを読み出すことができる。

ちなみに、キーを間違えると、nullが取得できます。

Drive27/35

1-4-3.消してみる

じゃ、今度はデータを消してみよう

```
NSUserDefaults * userDef = [[NSUserDefaults alloc] init];  
// [userDef setValue:@"あたた" forKey:@"俺の名前を言ってみる"]; //書き込み  
  
[userDef removeObjectForKey:@"俺の名前を言ってみる"]; //消す  
  
NSString * str = [userDef valueForKey:@"俺の名前を言ってみる"]; //読み出し  
NSLog(@"str_%@", str); //確認
```

→キーに対応したオブジェクトがuserDefに保存されているので、
以前保存しておいたパラメータを読み出すことができる。

ちなみに、キーを間違えると、nullが取得できます。

Drive28/35

1-4-3.忘れられない思い出って一般的に辛い事

簡単には、以上

どうやって消すの？

iPhoneのエミュレータからアプリケーションを削除すると消えます。

→逆に言うと、消えてしまいます。

キーはどうやって覚えとけばいいの？

→人力で。

保存出来るのはなんでもなの？

→なんでも、ではないんだな。

NSString、intなどの値は行けます。

自分のつくったオブジェクトとかは、簡単には出来ません。

(難しいけど出来ますという意味ですが。)

で、セーブ終わり！

Drive I / 35

3.Let's Rock (プログラムしましょう)

Drive I / 35

4. 「GitHub、君に決めた！」

Drive3 1/35

1-4-3.アプリケーションの保存を、Webサービス「GitHub」で行います
アドレスはここ。

<https://github.com/>

で、恐ろしい事に、各自授業用のアカウントを作ってください。
無料のやつがあるので、お薦めです。

Plans & Pricing - GitHub

/github.com/plans

gn in ドキュメント 見てるなう! Quix kissaki-blog Maps KISSAKI Flight KISSAKI ToDo_Google gb dev@twitter

Would you rather see this site in Japanese? (このサイトを日本語で利用しますか?) Yes (はい) No (いいえ)

github SOCIAL CODING

Pricing and Signup | Explore GitHub | Features | Blog | Login

Plans & Pricing

Join today and collaborate with the smartest developers in the world.

\$0/mo **Free for open source**
Unlimited public repositories and unlimited public collaborators

Create a free account

Drive32/35

1-4-3.アプリケーションの保存を、Webサービス「GitHub」で行います
いろいろ登録してね。



The screenshot shows a web browser window with the title "Sign up for GitHub - GitHub". The address bar shows the URL "git https://github.com/signup/free". The browser's bookmark bar includes links like "[KISSAKI.Inc] Sign in", "ドキュメント", "見てるなう!", "Quix", "kissaki-blog", "Maps", "KISSAKI Flight", and "KISSAKI ToDo_Goog". A black banner at the top of the page asks, "Would you rather see this site in Japanese? (このサイトを日本語で利用し)". Below this is the GitHub logo with the tagline "SOCIAL CODING" and a "Pricing" link. The main heading is "Sign up for GitHub". A yellow box highlights the pricing: "\$0/mo" and "You are signing up for the free plan". Below this, it states "The cost for this plan is \$0 per month. You can cancel, downgrade, or upgrade at any time." The section "Create your free personal account" contains two input fields: "Username" and "Email Address". To the right, there is a "facebook" button and a "You're" label. At the bottom right, there is a checked checkbox for "Email supp".

Sign up for GitHub - GitHub

git https://github.com/signup/free

[KISSAKI.Inc] Sign in ドキュメント 見てるなう! Quix kissaki-blog Maps KISSAKI Flight KISSAKI ToDo_Goog

Would you rather see this site in Japanese? (このサイトを日本語で利用し)

github
SOCIAL CODING

Pricing

Sign up for GitHub

\$0/mo You are signing up for the **free** plan
The cost for this plan is **\$0 per month**. You can cancel, downgrade, or upgrade at any time

Create your free personal account

Username

Email Address

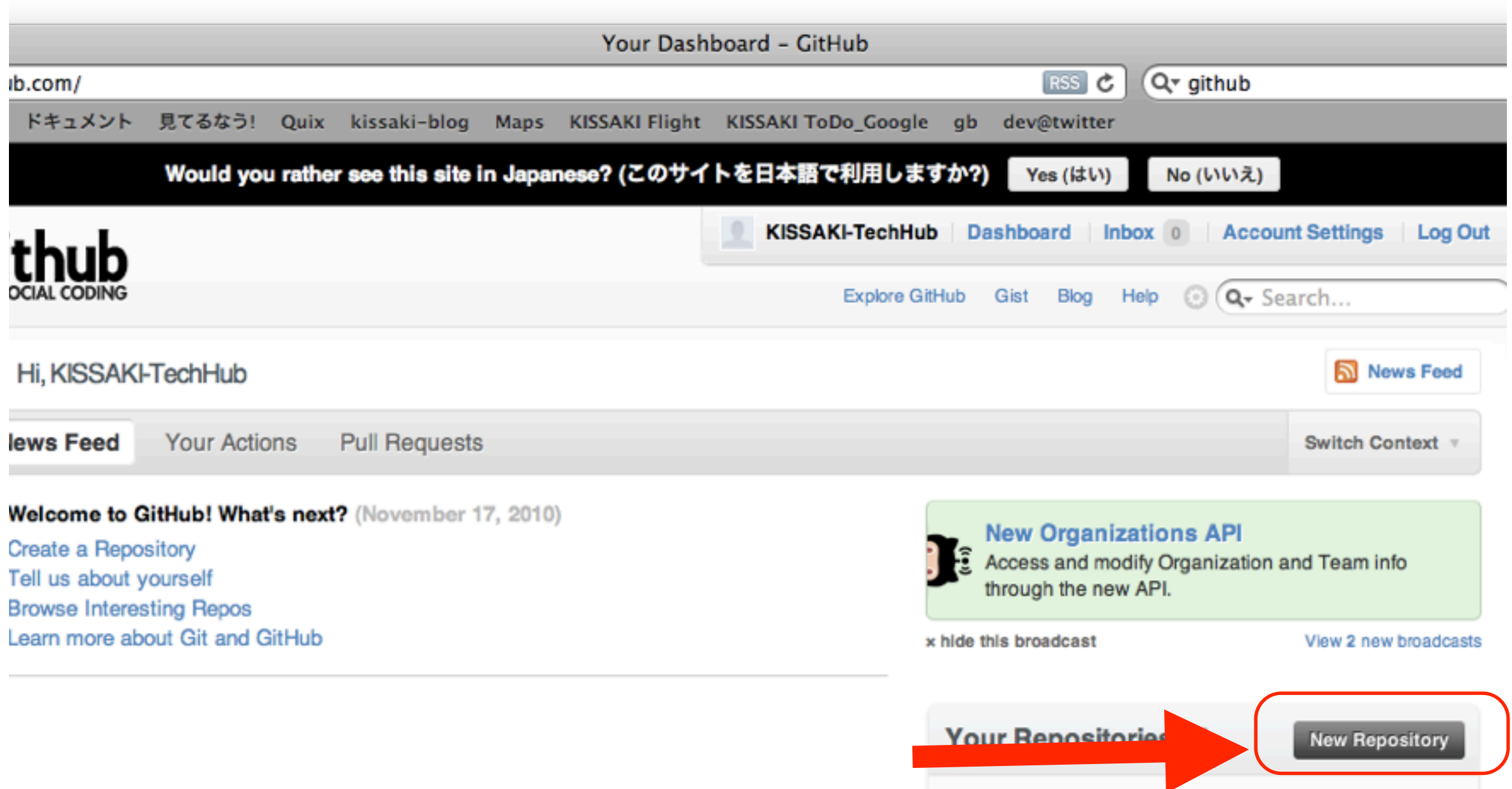
facebook

You're

✓ Email supp

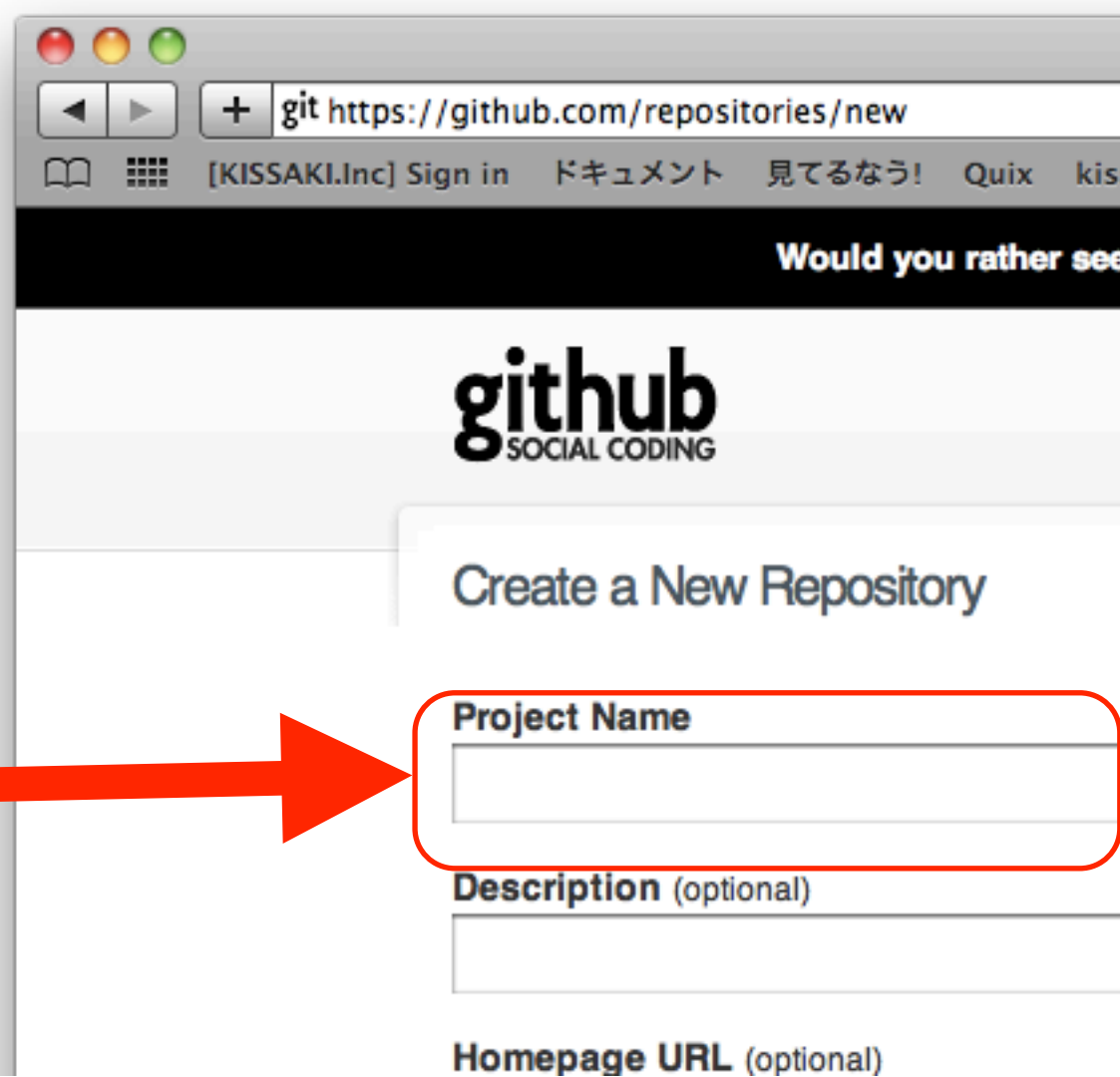
Drive33/35

1-4-3.アプリケーションの保存を、Webサービス「GitHub」で行います
で、リポジトリ（セーブ場所）をつくります



Drive34/35

1-4-3.アプリケーションの保存を、Webサービス「GitHub」で行います
で、リポジトリの名前は好きに決めてください。
アプリ名とかがお薦めです。



他の項目は、空欄でOKです。 で、CreateRepositoryボタンを押します。

Drive35/35

1-4-3.アプリケーションの保存を、Webサービス「GitHub」で行います
あとはまあ、画面に出てる内容を、Terminalというアプリから入力します。

アプリケーション > ユーティリティ > ターミナル



←こんなの。

＊ここから先の資料は、各人のセキュリティに関わる部分があるので、
別途送付します。

LookBack

- 1.プログラミング言語 座学 ファイナル
- 2.セーブ・セーバー・セーブスト
- 3.Let's Rock (プログラムしましょう)
- 4.「GitHub、君に決めた！」

プログラムの質問、提出はこっちな！！
自分の名前と宛名忘れんなよ！！



techhubteach@googlegroups.com



Continue?

さて、Objective-Cの講義が終わりました。
で、今回から後半戦に突入です。早いもんですね。

質問、解決出来ないエラー、こうしたいんだけどどうすればいいのか、などは Teachのアドレスまでメールすると、答えたりします。

次は来年だよ！！！！

次回、

第6回

「永久回廊(1)と永久回廊(2)と」