ReactiveCocoa

For Crafting Bespoke Artisanal State-Free Code.

March 7th, 2014

Terry Lewis

@tlewisii

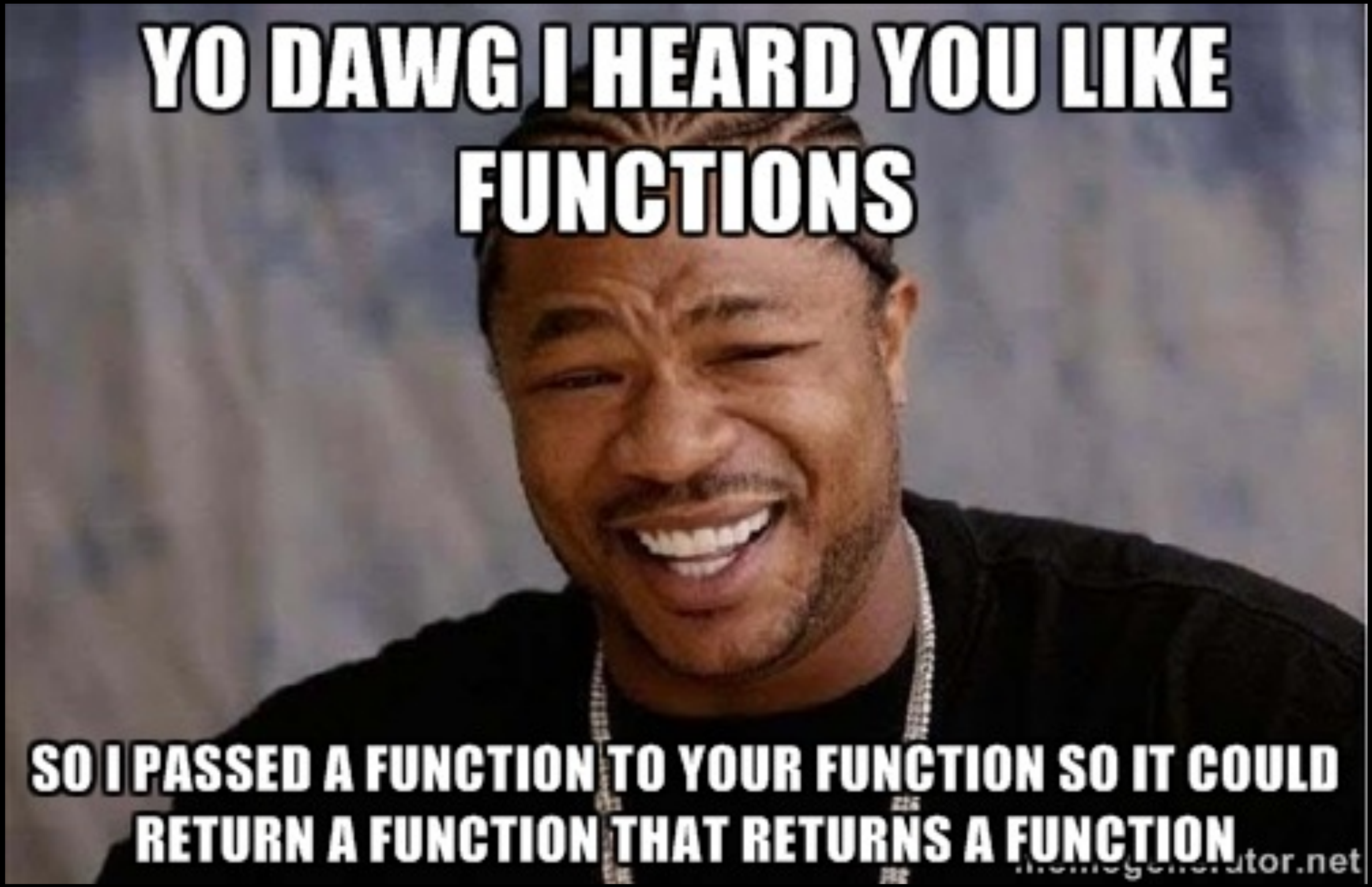
ReactiveCocoa <3 Functional Reactive Programming

What is Functional Reactive Programming (FRP), and why should you care?

Functional

A focus on functions, with functions being first class constructs in the language. Functions can be passed around and used just like any other value.

- Haskell
- Clojure
- Scala
- Elm
- Blocks in Objective-C


YO DAWG I HEARD YOU LIKE FUNCTIONS
SO I PASSED A FUNCTION TO YOUR FUNCTION SO IT COULD RETURN A FUNCTION THAT RETURNS A FUNCTION
memegenerator.net

```
f2 func2 = ^f0(f1 func1, id object) {
        return func1(object);
    };
```

Reactive

Change is automatically propagated in your app.

Think spreadsheets. Does Excel wait for you to manually reload it to perform calculations? Would you use it if that is how it behaved?

Declarative

Write what you want the program to do, rather than how it should do it.

Think Auto Layout. You give constraints and views compute their layout based on those constraints. You don’t care how it happens, just about the results.

Immutable

```
int x = 4;
    x = 5;
/* Back in my day,
compilers wouldn't let you do such unnatural things *\
```

Immutability reduces state, and reducing state leads to reduced complexity.

Reduced State and Side Effects

What is State? What are side effects?

State is anything that can change in place over time

- You lose past information. Once a variable has been changed, it's like the previous values never existed.
- Changes can come from any number of places. It's hard to look at stateful code and know exactly what will happen when — there's poor locality.
- Concurrency and asynchrony makes state management difficult, because you now have to coordinate changes across multiple execution points. Determinism is hard to achieve when there are multiple actors that may conflict with each other.

Justin Spahr-Summers
http://stackoverflow.com/a/19673276

Assignment is state.

Any time you use make an assignment or change a property, you are producing state.

```
int x = 4;
x = 5;
```

You just changed the state of the program.

State is difficult to reason about

BOOL flags here, other flags there, how many states can your program be in? More than you can reason about

Side Effects

A side effect is anything within a function that modifies or relies on state outside of the function.

```
- (NSUInteger)doSomethingStateful:(NSUInteger)integer {
  return self.globalVariable += integer;
}
```

Referential Transparency

A function is referentially transparent, or “pure” if it can be replaced by its value without changing the behavior of the program

That is, given the same input, the function will always produce the exact same output with no outside effects observable.

When your functions are pure, they become much easier to reason about.

```
- (NSUInteger)increment:(NSUInteger)integer {
    return integer + 1;
}
```

This function could be replaced by its value everywhere it is called, and the behavior of the program would not change.

Assignment and the `=` operator as we currently understand it.

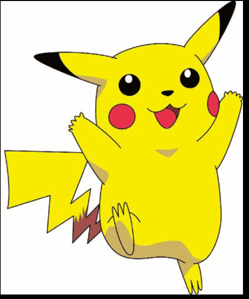
```
int x = 4;
int y = x;
x = 3; // y is still 4.
```

Assignment in FRP

```
int x = 4;
int y = x;
x = 3; // y is now 3
x = 2; // y is now 2
```

Assignment is assignment over time

when you say x = y, what you are really saying is that the value of x is equal to the value of y over time, rather than just its value at this particular moment. This is a key concept in ReactiveCocoa and FRP.



v0.1

Enough talk, show us the code!

We will be working through an Xcode project and going in steps. Whenever you see Pikachu, checkout the tag listed next to him.

the first step

- git clone https://github.com/tLewisII/RACTraining.git
- git checkout v0.1
- git checkout -b some_branch_name

RACSignal is the primary class that you will interact with.

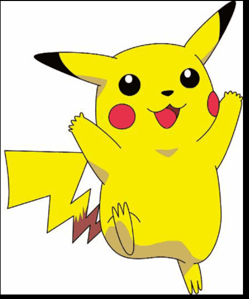
RACSignal is a push driven stream. Values are pushed onto a stream and then subscribers receive them.

Values from the textField are pushed onto the signal, and the subscriber receives the values as they are pushed.

```
[self.textField.rac_textSignal subscribeNext:^(NSString *text) {
        NSLog(@"%@", text)
}];
```

Similarly, values from RACObserve are pushed as well, and the subscribers receives them as they are pushed.

```
[RACObserve(self, pikachuString) subscribeNext:^(id x) {
        NSLog(@"%@", x);
    }];
```



v0.2

A subscriber is anything that registers to receive events from a signal.

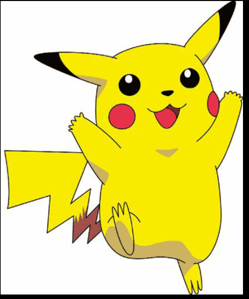
Signals are either cold or hot, and most signals are cold.

A cold signal will not do any work unless it is subscribed to, and it will execute its block for each subscription it receives.

[RACSignal createSignal:…
This creates a cold signal.
[RACSignal return:@“pikachu”];
This signal is also cold

a hot signal, by contrast, will immediately execute its given block, and will not execute it multiple times for multiple subscribers.

[RACSignal startEagerlyWithScheduler:…
This creates a hot signal signal.



v0.3

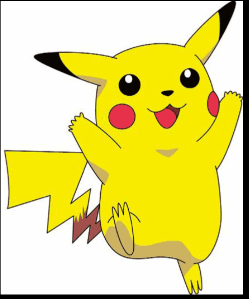
Scheduling

You can count on most signals to do their work synchronously, but scheduling background tasks is super easy with `-deliverOn:`.

Likewise, making sure UI work is done on the main thread is just as easy.

Remember, `-deliverOn:` affects the work of the returned signal, not the signal it was called on.

```
[[RACSignal createSignal:^RACDisposable *(id <RACSubscriber> subscriber) {
/// This work will not be affected by `-deliverOn:`
    }] deliverOn:[RACScheduler schedulerWithPriority:RACSchedulerPriorityBackground]];
```



v0.4

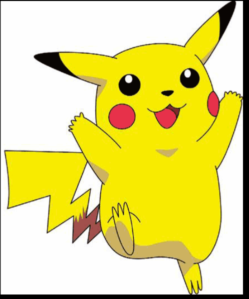
Transformations

RACSignal provides many functions that you may be familiar with from Ruby, such as:

- map:
- filter:
- flatten:
- take:

Signals are immutable, so all signal operators return a new signal, leaving the source signal untouched.

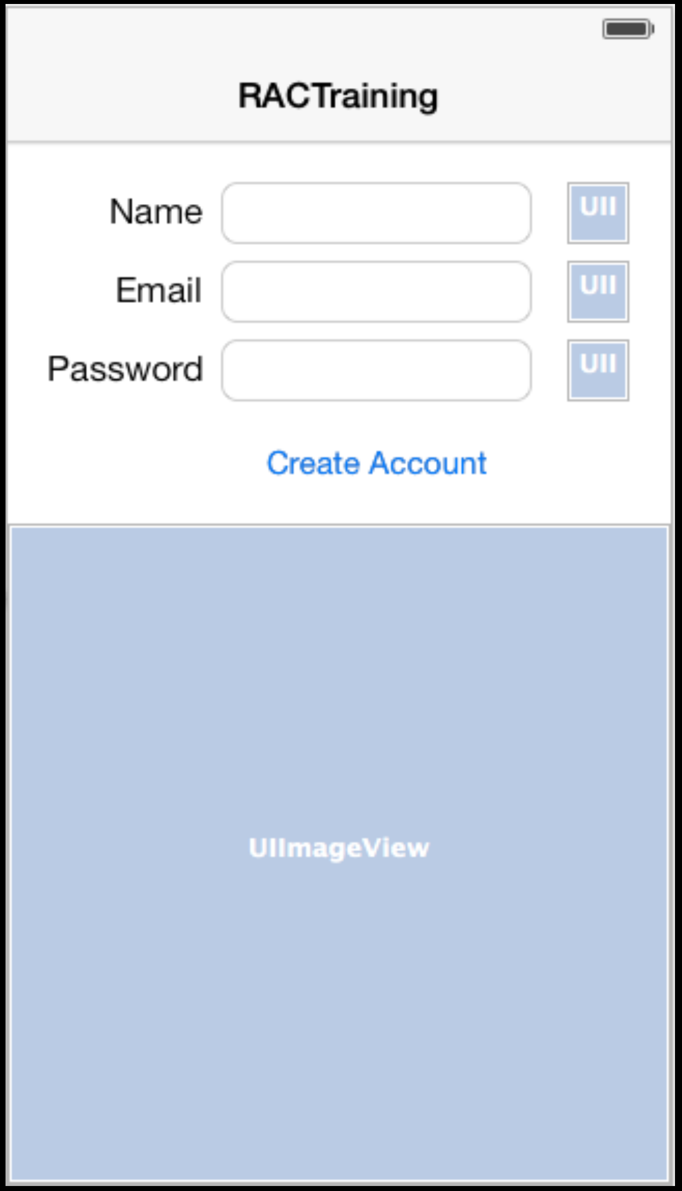
That means none of this `map!` nonsense for destructively operating on your signals.



v0.5

Now its your turn

We are going to build a simple login form in a few steps.
The UI and all the connections are already set up for you.



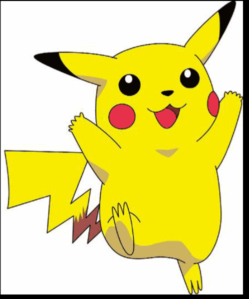
rac_textSignal will give you a signal that will send the value of the textField any time it changes

You can `map:` the signal and transform the text into a new value that you can use

The imageView next to each field should indicate in real time if the field is valid or invalid

So we want to transform the signal that we get from the text field into an image that we can send to the indicator imageView

```
RACSignal *nameSignal = [self.nameField.rac_textSignal map:^id(NSString *value) {
        return transformedValue;
    }];
```



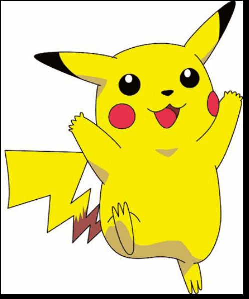
v0.6

All of the signals from the text fields need to be combined and then reduced to a single value that can be used to derive the state of the button

```
[RACSignal combineLatest:@[signal1, signal2, signal3]
  reduce:^(id obj2, id obj2, id obj3) {
    return reducedObject;
}];
```

The create account button should only be enabled when all of the text fields are valid and it should be disabled otherwise. This should also update in real time.

```
RACCommand *command = [[RACCommand alloc] initWithEnabled:correctnessSignal
                                              signalBlock:^RACSignal *(id input) {
        return workerSignal;
    }];
```



v0.7

When the create account button is pressed, it will make a network request and download an image from the given URL, and this image will be bound to the large imageView.

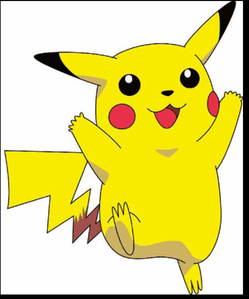
We also want to dismiss the keyboard when the button is pressed

```
[[self.createAccountButton rac_signalForControlEvents:UIControlEventTouchUpInside] subscribeNext:^(id x) {
   //dismiss the text fields.
}];
```

Finally, we want to show an activity spinner while we are executing the network request.

RACCommand has a `executing` property that can be used for this purpose.

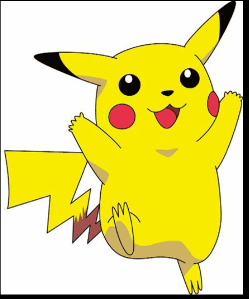
```
[command.executing subscribeNext:^(NSNumber *x) {
  // Do something here
    }];
```



v0.8

Finished!

We have a login form that presents real time information to the user about the actions that they need to take, that also prevents any incorrect actions from being taken.



v0.9

Moar fun

```
[self rac_signalForSelector:@selector(tableView:didSelectRowAtIndexPath:)]
```

Get a signal for a selector invocation, with its arguments sent in a RACTuple

Bro do you even lift?

```
[self rac_liftSelector:@selector(performSegueWithIdentifier:sender:)
withSignals:selectionSignal, [RACSignal return:nil], nil];
```

Lift a selector into the reactive world and it will be invoked whenever the given signals send their values

And thats it!

More resources

- https://github.com/ReactiveCocoa/ReactiveCocoa
- https://github.com/tLewisII/RACExample