Facebook POP 进阶指南

发布于：2014-07-04 14:43 阅读数：2396323963

> Facebook在发布了Paper之后，似乎还不满足于只是将其作为一个概念性产品，更进一步开源了其背后的动画引擎POP，此举大有三年前发布的iOS UI框架Three20的意味。

本文转自Kevin Blog

Facebook 在发布了 Paper 之后，似乎还不满足于只是将其作为一个概念性产品，更进一步开源了其背后的动画引擎 POP，此举大有三年前发布的 iOS UI 框架 Three20 的意味。而 POP 开源后也不负 Facebook 的厚望。

POP背后的开发者是 Kimon Tsinteris， Push Pop Press 的联合创始人，曾经在Apple担任高级工程师，并参与了 iPhone 和 iPad 上软件的研发(iPhone的指南针以及地图)。2011年的时候 Facebook 收购了他的公司，此后他便加入了 Facebook 负责 Facebook iOS 版本的开发。

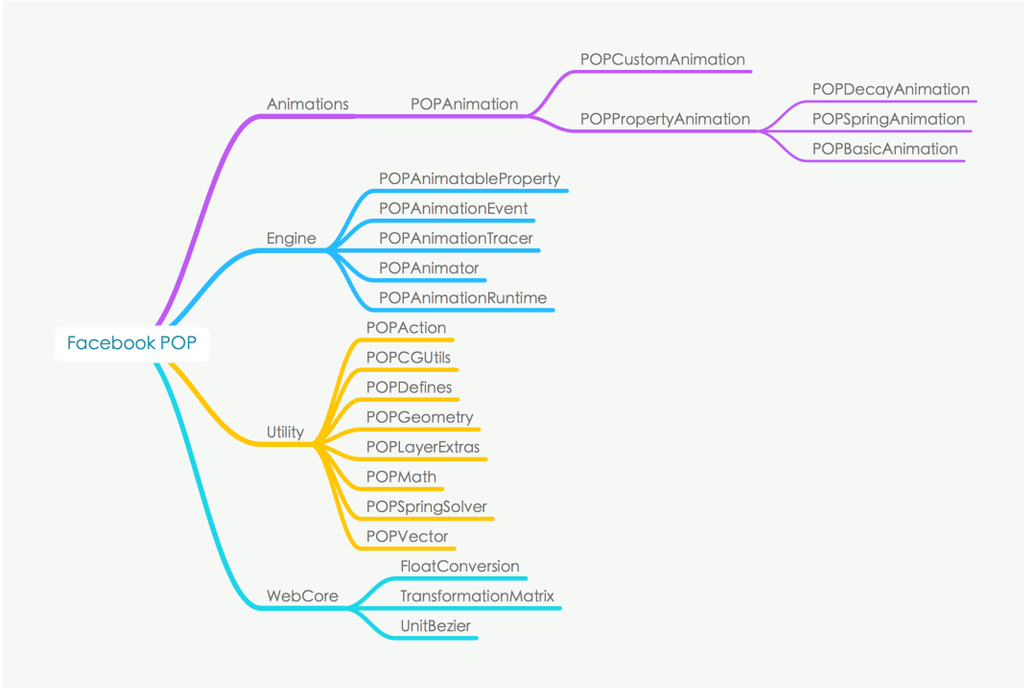
如果你打开 Push Pop Press 开发的 Al Gore 这款 App，你会发现交互和动画与Paper几乎如出一辙。对，他们都是 Kimon Tsinteris 开发的。

不满于 Apple 自身动画框架的单调，Push Pop Press 致力于创造一个逼真的、充满物理效应的体验。 POP 就是这个理念下最新一代的成果。

POP 使用 Objective-C++ 编写，Objective-C++ 是对 C++ 的扩展，就像 Objective-C 是 C 的扩展。而至于为什么他们用 Objective-C++ 而不是纯粹的 Objective-C，原因是他们更喜欢 Objective-C++ 的语法特性所提供的便利。

POP 的架构

POP 目前由四部分组成：1. Animations；2. Engine；3. Utility；4. WebCore。



POP 动画极为流畅，其秘密就在于这个 Engine 中的POPAnimator 里，POP 通过 CADisplayLink 高达 60 FPS 的特性，打造了一个游戏级的动画引擎。

CADisplayLink 是类似 NSTimer 的定时器，不同之处在于，NSTimer 用于我们定义任务的执行周期、资料的更新周期，他的执行受到 CPU 的阻塞影响，而 CADisplayLink 则用于定义画面的重绘、动画的演变，他的执行基于 frames 的间隔。

通过 CADisplayLink，Apple 允许你将 App 的重绘速度设定到和屏幕刷新频率一致，由此你可以获得非常流畅的交互动画，这项技术的应用在游戏中非常常见，著名的 Cocos-2D 也应用了这个重要的技术。

WebCore 里包含了一些从 Apple 的开源的网页渲染引擎里拿出的源文件，与 Utility 里的组件一并，提供了 POP 的各项复杂计算的基本支持。

由此通过 Engine、Utility、WebCore 三个基石，打造了Animations。

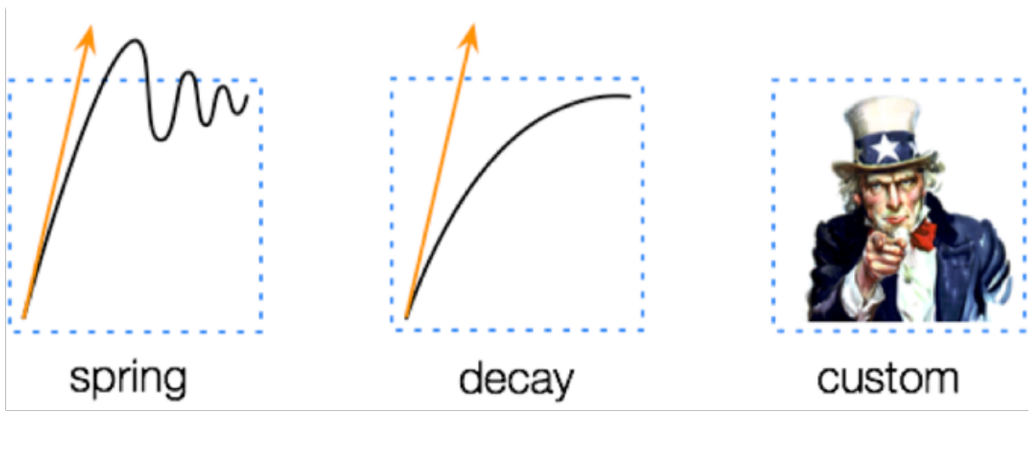
POPAnimation 有着和 CALayer 非常相似的 API。如果你知道 CALayer 的动画 API，那么你对下面的接口一定非常熟悉, 想必你一定开始迫不及待想试试 POP 了，我们现在就 Jump right in。

因为篇幅原因，下面的代码并不是完整代码，你可以到 https://github.com/kevinzhow/pop-handapp 获取我们的示例 App 。

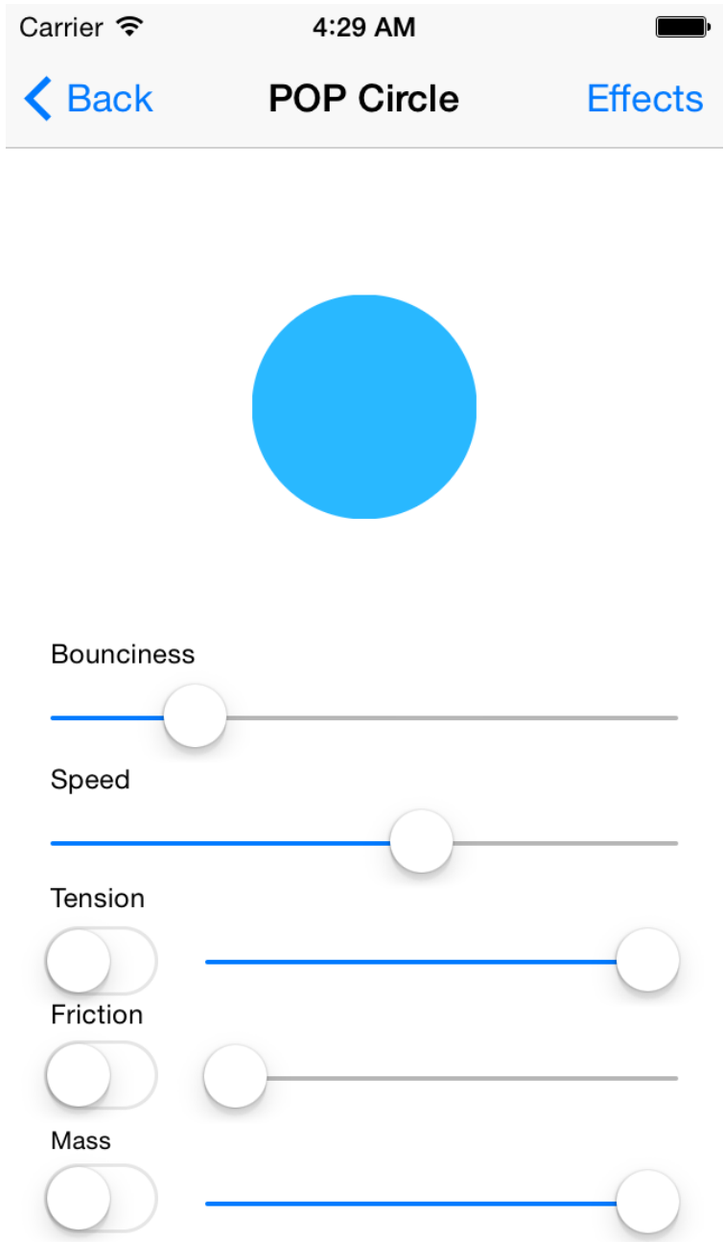
基本类型

Spring Animation

ease-in ease-out 这些可能你已经非常熟悉，这是动画的动作标配了，不过 POP 觉得只是这样显然太无聊，提供了两个非常不同于其的动画模式，第一个就是 Spring Animation。



Spring Animation 由诸多的复杂参数来控制，展现了一个非常风骚的姿势。



* Bounciness 反弹 – 影响动画作用的参数的变化幅度
* Speed 速度
* Tension 拉力 – 影响回弹力度以及速度
* Friction 摩擦力 – 如果开启，动画会不断重复，幅度逐渐削弱，直到停止。
* Mass 质量 – 细微的影响动画的回弹力度以及速度

Tension，Friction，Mass 这三个参数的作用很微妙，需要你在示例程序里去仔细体会。

使用 Spring Animation 的方式非常简单。

```
POPSpringAnimation *anim = [POPSpringAnimation animationWithPropertyNamed:kPOPLayerScaleXY];
anim.toValue = [NSValue valueWithCGPoint:CGPointMake(2.0, 2.0)];
anim.springBounciness = 4.0;
anim.springSpeed = 12.0;
anim.completionBlock = ^(POPAnimation *anim, BOOL finished) {
  if (finished) {NSLog(@"Animation finished!");}};
```

通过 [POPSpringAnimation animationWithPropertyNamed:kPOPLayerScaleXY] 我们创建了一个二维平面上分别沿着 X 和 Y 坐标轴进行缩放的动画。

因此我们要使用 toValue 来告诉 POP 我们希望分别缩放几倍，如果你不提供 fromValue，那么 POP 将默认从当前的大小为依据进行缩放。值得一提的是，toValue 这里的值要和动画作用的属性一样的结构。如果我们操作 bounds ，那么这里应该是 [NSValue valueWithCGRect:CGRectMake(0.0, 0.0, 200.0,400.0)]。

completionBlock 提供了一个 Callback，动画的执行过程会不断调用这个 block，finished 这个布尔变量可以用来做动画完成与否的判断。

最后我们使用 pop_addAnimation 来让动画开始生效，如果你想删除动画的话，那么你需要调用 pop_removeAllAnimations 。 与 iOS 自带的动画不同，如果你在动画的执行过程中删除了物体的动画，那么物体会停在动画状态的最后一个瞬间，而不是闪回开始前的状态。

Decay Animation

Decay Animation 就是 POP 提供的另外一个非常特别的动画，他实现了一个衰减的效果。这个动画有一个重要的参数 velocity（速率），一般并不用于物体的自发动画，而是与用户的交互共生。这个和 iOS7 引入的 UI Dynamic 非常相似，如果你想实现一些物理效果，这个也是非常不错的选择。

Decay 的动画没有 toValue 只有 fromValue，然后按照 velocity 来做衰减操作。如果我们想做一个刹车效果，那么应该是这样的。

```
POPDecayAnimation *anim = [POPDecayAnimation animWithPropertyNamed:kPOPLayerPositionX];
```

```
anim.velocity = @(100.0);
anim.fromValue =  @(25.0);
//anim.deceleration = 0.998;
anim.completionBlock = ^(POPAnimation *anim, BOOL finished) {
  if (finished) {NSLog(@"Stop!");}};
```

这个动画会使得物体从 X 坐标的点 25.0 开始按照速率 100点／s 做减速运动。 这里非常值得一提的是，velocity 也是必须和你操作的属性有相同的结构，如果你操作的是 bounds，想实现一个水滴滴到桌面的扩散效果，那么应该是 [NSValue valueWithCGRect:CGRectMake(0, 0,20.0, 20.0)]

如果 velocity 是负值，那么就会反向递减。

deceleration （负加速度） 是一个你会很少用到的值，默认是就是我们地球的 0.998，如果你开发给火星人用，那么这个值你使用 0.376 会更合适。

Property Animation & Basic Animation

POP 号称可以对物体的任何属性进行动画，其背后就是这个 Property Animation驱动。Spring Animation 和 Decay Animation 都是继承自这个类，接下来我们通过一个 Counting Label 的例子来展现这个神奇的能力。

与此同时我们也使用了 Basic Animation，经典的 ease－in－out 此刻发挥了重要的作用，因为我们并不需要计数器的数值进行回弹。

```
POPBasicAnimation *anim = [POPBasicAnimation animation];
anim.duration = 10.0;
anim.timingFunction = [CAMediaTimingFunction functionWithName:kCAMediaTimingFunction
EaseInEaseOut];
POPAnimatableProperty * prop = [POPAnimatableProperty propertyWithName:@"count" init
ializer:^(POPMutableAnimatableProperty *prop) { prop.readBlock = ^(id obj, CGFloat v
alues[]) {
    values[0] = [[obj description] floatValue];};
  prop.writeBlock = ^(id obj, const CGFloat values[]) {
    [obj setText:[NSString stringWithFormat:@"%.2f",values[0]]];};
  prop.threshold = 0.01;}];
anim.property = prop;
anim.fromValue = @(0.0);
anim.toValue = @(100.0);
```

POPBasicAnimation 的 timingFunction 我们定义了动画的方式，慢开慢停。随后通过 POPAnimatableProperty 定义了 POP 如何操作 Label 上的数值。

readBlock 中，obj 就是我们的 Label，values 这个是动画作用的属性数组，其值必须是 CGFloat ， 之前我们在 Decay Animation 中操作了 bounds

那么 values[0]，values1，values2，values3 就分别对应 CGRectMake(0, 0, 20.0, 20.0) 的 0, 0, 20.0, 20.0

这里我们只需要操作 Label 上显示的文字，所以只需要一个参数。通过 values[0] = [[obj description] floatValue] 我们告诉 POP 如何获取这个值。

相应的我们通过 [obj setText:[NSString stringWithFormat:@"%.2f",values[0]]] 告诉了 POP 如何改变 Label 的属性。

threshold 定义了动画的变化阀值，如果这里使用 1，那么我们就不会看到动画执行时候小数点后面的数字变化。 到此为止，我们的 Counting Label 就完成了，是不是超简单?

实战

PopUp & Decay Move

这个实例中我们介绍下如何将 Decay 动画和用户的操作结合起来，实现一个推冰壶的效果。

首先我们给我们的物体添加个 UIPanGestureRecognizer 的手势操作其，处理方式如下

```
case UIGestureRecognizerStateChanged: {
  [self.popCircle.layer pop_removeAllAnimations];
  CGPoint translation = [pan translationInView:self.view];
  CGPoint center = self.popCircle.center;
  center.x += translation.x;
  center.y += translation.y;
  self.popCircle.center = center;
  [pan setTranslation:CGPointZero inView:self.popCircle];
  break;
}
case UIGestureRecognizerStateEnded:
case UIGestureRecognizerStateCancelled: {
  CGPoint velocity = [pan velocityInView:self.view];
  [self addDecayPositionAnimationWithVelocity:velocity];
  break;
 }
```

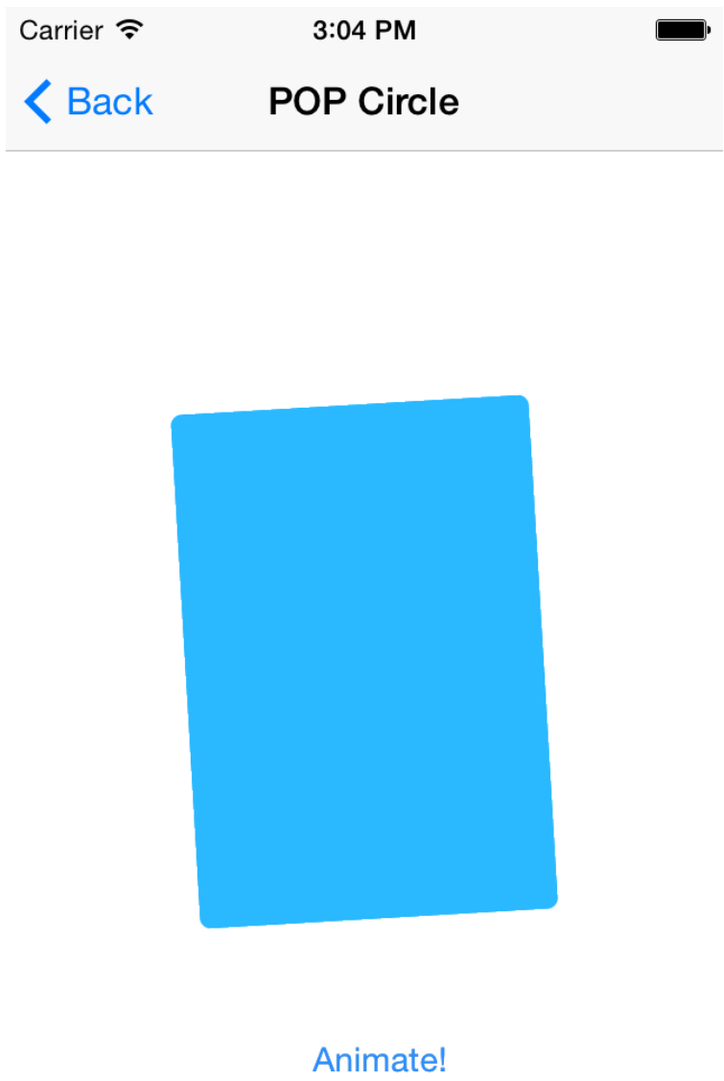
当用户触摸这个冰壶的时候，所有动画会立刻停止，然后跟随用户的手指移动。 通过 [pan velocityInView:self.view]; 我们获取了用户手指移动的速率然后在 addDecayPositionAnimationWithVelocity 中处理动画

```
POPDecayAnimation *anim = [POPDecayAnimation animationWithPropertyNamed:kPOPLayerPosition];
anim.velocity = [NSValue valueWithCGPoint:CGPointMake(velocity.x, velocity.y)];
```

当用户松开手之后，冰壶会依照地球的重力在低摩擦的状态下前进逐渐停止。如果想增大摩擦力，你可以把速率乘以一个摩擦系数。

Fly In

在这个实例中，我们介绍下如何结合两个动画。实现一个像 Path 的卡片飞入的效果。



同样保留了 Decay Move 的效果，你可以甩走这张卡片。

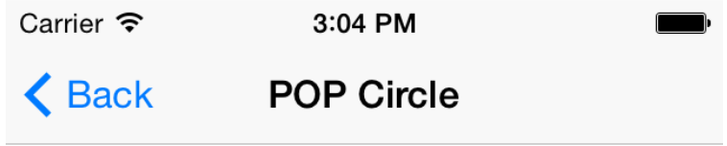
```
POPSpringAnimation *anim = [POPSpringAnimation animationWithPropertyNamed:kPOPLayerPositionY];
anim.fromValue = @-200;
anim.toValue = @(self.view.center.y);
```

```
04.
05.  POPBasicAnimation *opacityAnim = [POPBasicAnimation animationWithPropertyNamed:kPOPL
     ayerOpacity];
06.  opacityAnim.timingFunction = [CAMediaTimingFunction functionWithName:kCAMediaTimingF
     unctionEaseInEaseOut];
07.  opacityAnim.toValue = @1.0;
08.
09.  POPBasicAnimation *rotationAnim = [POPBasicAnimation animationWithPropertyNamed:kPOP
     LayerRotation];
10.  rotationAnim.timingFunction = [CAMediaTimingFunction functionWithName:kCAMediaTiming
     FunctionEaseInEaseOut];
11.  rotationAnim.beginTime = CACurrentMediaTime() + 0.1;
12.  rotationAnim.toValue = @(0);
```

首先把我们的冰壶变成卡片，旋转一点角度。这里需要注意的是，我们使用了 duration 来定义了 Basic Anim ation 的执行时间，beginTime 来定义了动画的开始时间。beginTime 接受的是一个以秒为单位的时间，所以我们使用了 CACurrentMediaTime() 获取了当前时间，然后加上了延迟时间。

Transform

这个实例是真的酷极了的效果，我们将实现一个用户点击后播放按钮转换为进度条容器的变形效果。



首先我们创建一个进度条，这个真是我最拿手的事情了。通过 lineCap lineWidth 我们调整进度条的样式，然后使用 UIBezierPath 定义了进度条的走向。

```
01.  CAShapeLayer *progressLayer = [CAShapeLayer layer];
02.  progressLayer.strokeColor = [UIColor colorWithWhite:1.0 alpha:0.98].CGColor;
03.  progressLayer.lineWidth = 26.0;
04.
05.  UIBezierPath *progressline = [UIBezierPath bezierPath];
06.  [progressline moveToPoint:CGPointMake(25.0, 25.0)];
07.  [progressline addLineToPoint:CGPointMake(700.0, 25.0)];
08.  progressLayer.path = progressline.CGPath;
09.
10.
11.  POPSpringAnimation *scaleAnim = [POPSpringAnimation animationWithPropertyNamed:kPOPL
     ayerScaleXY];
12.  scaleAnim.toValue = [NSValue valueWithCGPoint:CGPointMake(0.3, 0.3)];
13.
14.  POPSpringAnimation *boundsAnim = [POPSpringAnimation animationWithPropertyNamed:kPOP
     LayerBounds];
15.  boundsAnim.toValue = [NSValue valueWithCGRect:CGRectMake(0, 0, 800, 50)];
16.  boundsAnim.completionBlock = ^(POPAnimation *anim, BOOL finished) {
17.    if (finished) {
18.       UIGraphicsBeginImageContextWithOptions(self.popCircle.frame.size, NO, 0.0);
19.       POPBasicAnimation *progressBoundsAnim = [POPBasicAnimation animationWithPropert
     yNamed:kPOPShapeLayerStrokeEnd];
20.       progressBoundsAnim.timingFunction = [CAMediaTimingFunction functionWithName:kCA
     MediaTimingFunctionEaseInEaseOut];
21.       progressBoundsAnim.toValue = @1.0;
22.       progressBoundsAnim.completionBlock = ^(POPAnimation *anim, BOOL finished) {if (
     finished) {UIGraphicsEndImageContext();}};
23.       [progressLayer pop_addAnimation:progressBoundsAnim forKey:@"AnimateBounds"];
24.    }
25.   };
```

首先是一起进行的 scale 和 bounds 的变化效果，播放按钮将缩小然后改变外形成为进度条的容器，在变形

结束后，我们触发进度条的动画。

这里我们使用 UIGraphicsBeginImageContextWithOptions(self.popCircle.frame.size, NO, 0.0); 开启了绘画上下文，动画结束后使用 UIGraphicsEndImageContext(); 清空了绘画上下文。这个主要是影响了画板的大小。

这里我们没有使用 UIGraphicsBeginImageContext() 而是使用 UIGraphicsBeginImageContextWithOptions() 以此获取一个更清晰的绘图效果。

值得关注的 POP 周边

POP-HandApp 这就是本文的示例App，包含了大量动画的操作方法和上述介绍的实例。

AGGeometryKit-POP 通过 POP 对图片进行变形操作，非常酷。

POP-MCAnimate POP 的一个封装，可以让你更方便的使用 POP。

Rebound POP 的 Android 部分实现，主要是 Spring 的效果，移植自 Facebook 的rebound-js。

结语

POP 是一个新的里程碑，通过 POP，动画的开发门槛大大降低，并且实现了丰富的属性操作，其倡导的可中断式动画交互会革命性也值得我们仔细研究体会，想必不久就会涌现大量富有活力的 App ，感谢 Facebook ，感谢开源。 Long live Opensource.

静止的枫叶

不错，值得学习！

2015年3月11日 回复 顶 转发

爱奇趣

爱奇趣网http://www.iqiqu.net/? 路过留个言！

2015年8月24日 回复 顶 转发

社交帐号登录: 微信 微博 QQ 人人 更多»

说点什么吧...

发布

www.cocoachina.com正在使用多说