       Animated库可以让开发者非常容易并且非常高效的性能实现各种的动画以及交互的方式。使用Animated的时候，我们只需要关注设置动画的实现和结束即可，然后在里边设置一个动画可配置的函数。间接着通过start/stop的方法来控制动画按照顺序执行。例如下面就是一个在加载的时候带有比较简单的弹跳动画的效果实例:  
         上面代码，其实bounceValue作为state的一部分在构造函数中进行初始化，然后复制给图片的缩放比例。在动画不断执行的时候，该数值会发生改变然后重新设置缩放大小。当组件被挂载的时候，该缩放值设置到1.5，然后在bounceValue值基础上面运行弹跳动画(spring),在动画运行过程中会重新刷新缩放值，同时图片的大小缩放比例也会发生变化了。这种方法比起调用setState方法并且重新刷新组件效率来的高。因为这一切配置都是直接声明配置的，当然我们还可以进一步进行优化动画效果。例如可以配置顺序执行的动画，然后整体动画效果会跟着我们的配置进行相应的执行。

2.0.根据官方的文档整理梳理了一下Animated库需要了解并且掌握的相关内容图如下:

       上面是Animated库的相关知识点，接下来我们会一块块的进行详细讲解分析，同时在本专题第二篇会用大量的实例进行演示使用

2.1.核心API(Core API)

         我在使用动画的时候大部分需要使用的模块方法都来自Animated模块，该包括两个值类型:①.Value代表单一值,ValueXY代表向量值。②.三个动画类型:spring,decay和timing。③.三个组件类型:View,Text和Image，并且另外通过Animated.createAnimatedComponent自定义创建的动画组件。

通过这三种动画类型我们几乎可以创建任何需要的动画效果:

2.1.1.spring:该为弹跳类型的动画

①.friction:  bounciness值,摩擦力值  默认为7

②.tension:  弹跳的速度值  默认为40

2.1.2.decay:带有加速度值的动画，类似于正弦值

①.velocity:初始速度,必须要填写

②.deceleration:速度减小的比例，加速度。默认为0.997

2.1.3.timing:带有时间的渐变动画

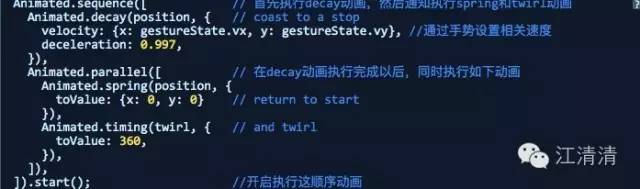
①.duration:设置动画持续的时间(单位为毫秒),默认为500ms

②.easing:动画效果的函数，大家可以查看Easing模块查询更多的预定义的函数。iOS平台的默认动画为Easing.inOut(Easing.ease)

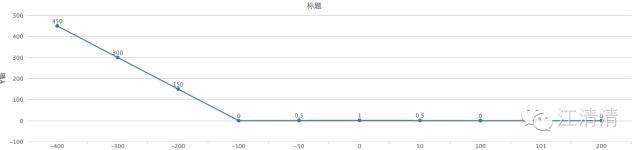
③.delay: 动画执行延迟时间(单位:毫秒).默认为0ms

         动画通过调用start方法开始执行，start方法可以设置一个回调方法，当动画执行完毕会进行调用该方法。如果动画是正常播放完毕的，那么该回调方法会被执行并且传入参数{finished:true}。但是如果动画是在正常播放完毕之前调用了stop来进行停止的话，那么该会回调传入{finished:false}。

2.2.组合动画(Composing Animations)

        如果我们有多个动画，那么可以进行使用单个动画组合起来进行执行。具体可以通过parallel(同时执行),sequence(顺序执行),以及stagger和delay来进行组合使用动画。这个几个类型都可以传入一个需要执行的动画的数组，然后自动start/stop来进行开始动画或者停止动画。具体可以看一下官方的例子:  
        默认情况下,如果一个动画被停止或者中断了，那么在该组合动画中的其他动画也停止了。不过Parallel同时执行动画中有一个stopTogether可选属性，如果设置这个属性为false的话，那么就可以禁止自动停止啦。

2.3.插值动画(interpolate)

        Animated库API还提供一个非常强大的功能是interpolate插值动画效果。该允许一个输入的区间范围映射到另外一个输入的区间范围。例如:一个简单的实例0-1的区间映射到0-100的区间范围:  
       interpolate还支持多段区间，该用来定义一些静态区间。例如:当我们输入为-300的时候取值相反值。然后在输入到-100的时候变成0，当输入接着0 的时候变成1，接着输入一直到100的时候又回到0。最终形成 的静态区间就为0。大于任何大于100的输入都返回0.具体事例如下:  
类似的映射例子如下:  


         interpolation还支持任意形式的渐变方法，很多类型已经Easing类中定义了，例如:二次函数,指数函数,贝塞尔曲线等step以及bounce。interpolation还可以设置区间outputRnage的输出值边界。可以设置extrapolate,extrapolateLeft或者extraplateRight可选属性。默认值为extend(允许超出),但是我们可以设置clamp属性来避免值超出outputRange的最大边界值。

[注意].这一插值函数小节大家可能也有小抽象，其实如果没有用过的人会感觉到非常的抽象，如果大家以前学过离散数学的话，可能理解起来会稍微好一点，不过也不用太担心，我会在后面的博客中专门讲解一下插值函数相关的知识点以及实例的。

2.4.追踪动态值(Tracking Dynamic Values)

        在设置的动画中所设定的值可以通过追踪别的值进行获取到。我们只需要不设置toValue为一个普通固定的数组就可以了。例如我们想实现聊天头像的闪动的效果。又例如我们需要timing动画快速的在后面实现。  
        ValueXY是可以非常方便的处理2D交互，例如旋转或者拖拽动画。该ValueXY是简单的封装了两个Animated.Value实例以及一些可以去调用的方法。在很多情况下，我们可以使用ValueXY替换Value进行使用。上面的例子中，leader和follower都可以同时为ValueXY类型，这样x和y的值我们都可以获取到了。

2.5.输入事件(Input Events)

        Animated.event在Animated API中和手势输入模块相关。该允许手势事件以及其他事件操作。该通过一个结构化得数组进行拆分显示复杂的事件对象。该方法参数是一个数组可以映射输入多个事件的对象。我们来看下面的一个例子:我们把event.nativeEvent.contentOffset.x的值赋值到scrollX变量中(event一般为回调方法的第一个参数)。同时取出手势状态的gestureState.dx和gestureState.dy的值赋值到pan.x何pan.y变量中(gestureState通常为PanResponder回调方法中的第二个参数)。来我们看代码示例:

2.6.动画运行动态值

          经过查看文档，也没有发现特定的方法让我们可以主动去获取在动画执行过程中的动态值信息。但是如果我们需要在JavaScript中可以监听到当前动画的响应值，可以有以下两种方法获取到:

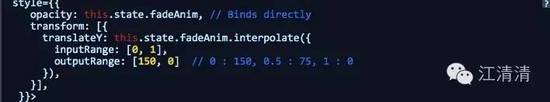
1. spring.stopAnimation(callback)  该方法当在停止动画的时候会把最终的响应值通过参数中的回调方法进行回调出来。该在处理手势的时候非常有用。
2. spring.addListener(callback)  该当动画在运行过程中会异步回调callback方法，返回一个最近的动画响应值。该方法在触发状态改变的时候非常有用。不过该方法不会特别的灵敏。

#### 2.1.Animated.Value介绍

最基本的一个动画使用方式是创建一个Animated.Value,将该动画绑定到一个或者多个样式属性到动画组件中，然后通过开启动画运行。例如 Animated.timing，或者采用Animated.event绑定到拖动或者滚动的手势中。除了绑定到样式中，Animated.Value也 可以绑定到属性中(props)，同样也可以加入插值动画。下面一个很简单的例子进行演示视图在加载的时候淡入显示:    
大 家请注意，只有声明了可以进行动画设置的组件才能有动画效果，例如:View,Text和Image这三个组件都可以提供动画效果。当然了，如果大家看过 之前的Animated基础篇的童鞋可以也了解到，我们同样可以采用createAnimatedComponent方法创建一个可以有动画效果的自定义 组件。这些特殊的组件可以绑定动画值到属性中，然后每一帧通过原生的方法去刷新。这样可以避免每次都去渲染或者同步过程中的开销。同时这些组件默认在卸载 的时候会做一下释放清理工作已达到安全的使用。

动画库具备很强大可配置性，例如可以配置自定义或者预先定义的过渡方法，延迟加载，动画时间，衰减幅度，弹跳度等根据动画类型不同配置可用的属性参数。

一个Animated.Value可以绑定驱动任何数量的动画属性，并且每一个属性可以通过插值函数来运行。插值通过两个区间进行映射，通过我们 一般使用一个线性的插值函数，但是也可以使用其他的过渡函数。默认情况下，该当输入的区间超过范围时候，该也会进行相应的转换工作，不过我们可以设置把输 出设置在一个可约束的范围之内。

下面我们来看一个实例，我们需要通过Animated.Value的值从0变化到1，组件的位置从150px移动到0px，同时透明度从0变化到1。我们可以非常方便的修改style就可以实现。如下代码:  看过之前的基础的童鞋知道，动画还可以进行通过各种方式进行组合形成比较复杂的动画。主要是通过sequence和parallel方法。同时可以通过把toValue设置成另外的Animated.Value来产生一个动画序列效果。

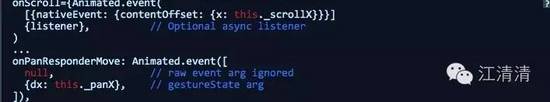
#### 2.2.Animated.ValueXY介绍

Animated.ValueXY可以用来处理一些2D动画效果，例如滑动。当然还有一些其他的方法setOffset和getLayout可以 用来帮助实现一些交互的动作，例如:拖拽操作。大家可以查看官方写的AnimationExample.js文件查看更多的使用例子。

[注意].Animated是被设计成可以完成序列化的，这样可以脱离常规的JavaScript的事件循环，以一种高效性能的方式运行。所以整 体设计出来的API可能和同步的系统动画相比有点奇怪。不过我们可以使用Animated.Value.addListener进行监听动画执行过来中的 一些状态，不过在使用过程中需要注意一下性能问题，尤其在以后的版本中使用。

#### 2.3.基本使用方法

* decay(value,config) 静态方法,传入一个初始速度值以及衰减值。动画从初始速度慢慢衰减到0.
* timing(value,config)静态方法,该动画传入一个值，根据过渡曲线函数变化。Easing模块已经定义很多不同的过渡曲线方法，当然也可以自己自定义
* spring(value,config)静态方法,创建一个基于Rebound和Origami实现的Spring动画。该值会将当前动画运行的状态值自动更新到toValue属性中，以确保动画的连贯性。可以链式调用。
* add(a,b)  静态方法,将两个动画值相加，创建一个新的动画值。
* multiply(a,b) 静态方法,将两个动画值进行相乘，创建一个新的动画值
* modulo(a,modulus) 静态方法，进行对参数一的动画值取模(非负值)，创建一个新的动画值
* delay(time)  静态方法，在给定的延迟时间之后执行动画
* sequence(animations) 静态方法，该按照顺序执行一组动画，该需要等待一个动画完成以后才会继续执行下一个动画。如果当前的动画被打断终止了，那么就不会执行后边的动画了。
* parallel(animations,config?)  静态方法，同时执行一组动画，默认情况下，如果其中有任一动画被终止了，那么其余的动画也会被停止。不过我们可以通过stopTogether来改变设置。
* stagger(time,animations) 静态方法，执行一组动画，有可能里边的动画是同时执行。不过会有指定时间的延迟。
* event(argMapping,config?) 静态方法  响应事件值，如下看一下使用方法



* createAnimatedComponent(Component) 静态方法 ，使得任何React的组件都可以设置动画效果，例如创建Animated.View等等

#### 2.4.基本属性

* Value:AnimatedValue   数值的类，用于动画效果。通常我们使用new Animated.Value(0)，进行初始化动画
* ValueXY:AnimatedValueXY  来用实现2D动画效果，例如拖拽操作等

AnimatedValue类

* 该值用来驱动动画执行，一个Animated.Value可以用一种顺序的方法进行执行多种属性，不过在同一时间只能执行一种效果。通过开启一个新的动画或者调用setValue方法会停止之前任何的动画然后开始新的动画效果。
* 3.1.重要方法
* constructor(value)  构造方法，初始化一个值
* setValue(value)   直接设置动画值，该会停止任何正在进行中的动画，然后更新所有绑定的属性
* setOffset(value)  设置偏移量，接下来无论使用setValue,一个动画或者Animated.event，都会加上这个值。常用在手势拖动操作中。
* flattenOffset()   该用来把相对值合并到值里，然后相对值设置为0，最终输出的值不会发生变化。常常在拖动操作结束以后调用。
* addListener(callback)  添加监听器，用于监听动画执行过程中的值的变化。
* removeListener(id)   删除指定的监听器
* removeAllListeners()  删除所有的监听器
* stopAnimation(callback?)  进行停止任何正在运行的动画或者追踪器，最终callback会被调用，该参数为动画结束的时候那个最终值。
* interpolate(config)   在更新属性之前对值进行插值函数映射 ：例如 映射0-1到0-10
* animate(animation,callback)   私有方法，一般在自定义动画类中可能会使用到
* stopTracking()  私有方法
* Track(tracking)  私有方法

AnimatedValueXY类

用设置驱动2D动画的运行，例如滑动操作等。该使用的API和普通的Animated.Value几乎一模一样，只不过该包含两个Animated.Value值的复杂结构。实例如下: 

#### 4.1.重要方法

* constructor(value?)
* setValue(value)
* setOffset(offset)
* flattenOffset()
* stopAnimation(callback?)
* addListener(callback)
* removeListener(id)
* getLayout()

用于样式中，将{x,y}的形式转换成{left,top}的模式，例如: style={this.state.anim.getLayout()}

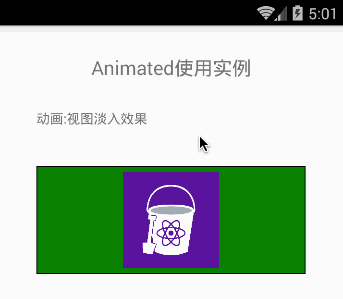
* getTranslateTransform()

用于将{x,y}的组合形式转换成可以用于平移变化的模式,例如:style={{transform: this.state.anim.getTranslateTransform()}}

实战实例

上面我们已经把用于动画效果实现的主要Animated.Value和Animated.ValueXY讲解了一遍，以及配合上一讲的Animated动画基础篇的知识点，下面我们通过几个实例来具体演示一下动画操作实现。

5.1.首先我们来看一下视图的淡入的效果，核心代码如下:  运行效果



#### 5.2.动画在移动过程中(旋转,平移,缩放)，加入插值函数。核心代码如下:

 运行效果:



#### 5.3.组合动画演示,这边是采用顺序动画,核心代码如下:

 运行效果:



最后说两句