链式编程原理与应用(Autolayout)

1. 简述：作为一个 iOS 程序员基本上都应该接触过 Masonry 这个自动布局库。这个库能够帮助程序员极大程度的简化自动布局的代码。使用这个库让我感到惊叹的不是 如何能够将较为复杂的传统自动布局写法精简到如此程度，而是精简后的代码的书 写方式

   make.left.right.bottom.mas_equalTo(0.f); 这种写法在做到简化的同时，通过点(.)调用的方式，将代码连接成一行，大大增加 了代码的可读性，这就是本篇要提到的链式编程。关于链式编程具体是种什么编程 思想，这种概念性的东西，请自行百度，这里不多做介绍。 本人在理解了链式编程的基础上，也封装了一个项目使用的轻量级自动布局库 (MYAutolayout，链接：https://github.com/cwn152522/MYAutoLayout)，很好用噢， 下面会进行介绍。

2.实现：

在百科了链式编程思想后，再谈谈我对链式编程在 ios 端实现过程的一个理解: 以 MYAutolayout使用为例：

```
[self.progressView cwn_makeConstraints:^(UIView *maker) {
    maker.leftToSuper(20).rightToSuper(20).height(10).bottomTo(weakSelf.blueView, 1, 10);
}];
```

对于 topLine，我们对其设的约束有：距父视图左边 20，居父视图右边 20，居 blueView 顶部 10，高度 10。大家可以看到，这些约束的设置，在一行代码就完成了。

(1)首先，progressView 初始化完毕，需要做约束，我们调 cwn_makeConstraint: 方法，这个方法带了个block参数返回一个maker对象，然后之后使用maker来做约 束，maker 其实就是 progressView(好像很厉害的样子，其实只是仿照 Masonry 自动布 局库的写法，把 progressView 直接返回而已)；然后我们把注意力集中在链式编程代 码上：

maker.leftToSuper(20).rightToSuper(20). ..

(2) leftToSuper(20)是一个方法，这个方法好像和平时写的 oc 方法不太一样？传 入参数不是用：？其实，方法传入参数并不是 20，压根这个方法就没有参数！链式编 程的方法只能是get方法。大概方法流程是这样的:调用get方法获取一个返回值为

UIView的block，调用这个block，传入参数20，block进行逻辑处理，比如调用系统 自动布局api，设置progressView距父视图左边20，设置完之后，把maker，也就是progressView 返回，因为返回了 maker 对象，所以可以接着执行下一个方法 rightToSuper，然后一直执行下去，就是这么回事。

(3)以下为链式编程函数写法：

```
- (UIView *(^)(CGFloat))leftToSuper{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(CGFloat) = ^(CGFloat constant){
        [weakSelf setLastConstraint:[weakSelf setLayoutLeftFromSuperViewWithConstant:constant]];
        return weakSelf;
    };
    return block;
}
```

看了(2)的解释后来看上图，关于链式编程实现是不是一目了然了？

3. 以下附上 MYAutolayout 库的封装，大家请确保链式编程思想、实现原理都懂了后来看，不然，可能有点吃力噢^_^

(1) MYAutolayout主要实现文件为UIView的分类，如下

UIView+CWNView.h

```
*
 @author 陈伟南, 17-04-11 22:05:55

 引入链式编程思想，进一步简化autolayout代码


 @author 陈伟南, 17-04-28 13:58:34

 新增frame布局的适配方法，目前提供相对父布局的便捷适配


 @note  关于constant符号说明:

        (1)所有方法传入的constant均传正数即可，部分方法内部需使用负数时会自动转换。
        (2)当外界需对某个约束进行更新时(改变约束的constant)，这时候就得注意正负值了。
        (3)constant正负取决于参照视图和自身的位置关系，比如:a.right = b.left + constant，这个约束表示a的右边距离b的左边constant处。如果你希望a和b间关系是相离，那么constant得为
 数，因为如果是正数的话a和b就相交了。
/



interface UIView (CWNView)

pragma mark 布局操作器获取方法，在block里调用具体布局方法进行布局

*
 autolayout布局操作器获取方法

 @ param maker   待约束视图，即自身
/
void)cwn_makeConstraints:(void (^)(UIView *maker))block;
```

```
    /**
     * frame布局适配操作器获取方法
     *
     * @ param maker    待适配视图，即自身
     */
    - (void)cwn_makeShiPeis:(void (^)(UIView *maker))block;


    #pragma mark 具体约束设置方法(分新旧两套)，根据个人喜好，自行选择

    #pragma mark ----------------------------------新版本链式编程----------------------------------
    #pragma mark --------------------autolayout布局----------------------------
    /**
     * @note 适用于布局内控件间有特定的位置关系且所有控件大小不需适配的场景，比如顶部导航的封装(控件大小固定、中间文本居中、左按钮居左、右按钮居右)、高度固定的自定义cell布局等
     */

    /**
     * 最新创建的一个约束获取方法
     *
     * @note 这个约束只记录以下方法执行结束时产生的约束，是个临时值
     * @note 用途：动态更新，需先定义变量进行存储
     */
    @property (strong, nonatomic) NSLayoutConstraint *lastConstraint;

    /**
     *  控件相对父视图约束设置方法
     *
     * @ param constant  上下左右相对父视图的距离
     */
    - (UIView *(^)(CGFloat constant))topToSuper;
    - (UIView *(^)(CGFloat constant))leftToSuper;
    - (UIView *(^)(CGFloat constant))rightToSuper;
    - (UIView *(^)(CGFloat constant))bottomToSuper;
/**
 *  控件间相对约束设置方法
 *
 * @ param targetView    参照视图
 * @ param multiplier   比例
 * @ param constant     常数
 * @ note   setLayoutLeft:方法相对的是参照视图的Right，其他方法同理
 */
- (UIView *(^)(UIView *targetView, CGFloat multiplier, CGFloat constant))topTo;
- (UIView *(^)(UIView *targetView, CGFloat multiplier, CGFloat constant))leftTo;
- (UIView *(^)(UIView *targetView, CGFloat multiplier, CGFloat constant))rightTo;
- (UIView *(^)(UIView *targetView, CGFloat multiplier, CGFloat constant))bottomTo;

/**
 *  控件宽高的约束设置方法
 *
 * @ param targetView     参照视图
 * @ param multiplier   比例
 * @ param constant    常数
 */
- (UIView *(^)(CGFloat constant))width;
- (UIView *(^)(CGFloat constant))height;
- (UIView *(^)(UIView *targetView, CGFloat multiplier, CGFloat constant))widthTo;
- (UIView *(^)(UIView *targetView, CGFloat multiplier, CGFloat constant))heightTo;

/**
 *  控件中心对齐约束设置方法
 *
 * @ param targetView    参照视图
 * @ param constant   常数
 */
- (UIView *(^)(CGFloat constant))centerXtoSuper;
- (UIView *(^)(CGFloat constant))centerYtoSuper;
- (UIView *(^)(UIView *targetView, CGFloat constant))centerXto;
- (UIView *(^)(UIView *targetView, CGFloat constant))centerYto;


#pragma mark ----------------------frame适配----------------------------
/**
 * frame相对父布局适配
 *
 * @note 将指定view及其subview(iphone6下进行frame布局的)的frame参数均乘以适配参数(当前屏幕的宽度和iphone6的宽度比)进行适配
 * @note 适用于布局内控件相对父控件有特定的位置关系且所有控件大小均需适配的场景，比如中间弹窗提示类情景、高度不固定的自定义cell布局等
 * @note 建议storyboard或xib下用autolayout布局完成(按ui设计调好)后，禁用autolayout(移除所有约束)，然后代码中通过执行以下两个接口，确保所有控件的frame都适配好。
 */

/**
 * 相对父布局适配之父视图适配
 */
- (UIView * (^)())shiPeiSelf;

/**
 * 相对父布局适配之子视图frame适配
 */
- (UIView *(^)())shiPeiSubViews;
```

```
#pragma mark ----------------------------------旧版本----------------------------------------
#pragma mark ---------------------autolayout布局----------------------------
/**
 * @note 适用于布局内控件间有特定的位置关系且所有控件大小不需适配的场景，比如顶部导航的封装(控件大小固定、中间文本居中、左按钮居左、右按钮居右)、高度固定的自定义cell布局等
 */

- (NSLayoutConstraint *)setLayoutLeftFromSuperViewWithConstant:(CGFloat)c;
- (NSLayoutConstraint *)setLayoutTopFromSuperViewWithConstant:(CGFloat)c;
- (NSLayoutConstraint *)setLayoutRightFromSuperViewWithConstant:(CGFloat)neg_c;
- (NSLayoutConstraint *)setLayoutBottomFromSuperViewWithConstant:(CGFloat)neg_c;

- (NSLayoutConstraint *)setLayoutLeft:(UIView *)targetView multiplier:(CGFloat)multiplier constant:(CGFloat)c;
- (NSLayoutConstraint *)setLayoutTop:(UIView *)targetView multiplier:(CGFloat)multiplier constant:(CGFloat)c;
- (NSLayoutConstraint *)setLayoutRight:(UIView *)targetView multiplier:(CGFloat)multiplier constant:(CGFloat)neg_c;
- (NSLayoutConstraint *)setLayoutBottom:(UIView *)targetView multiplier:(CGFloat)multiplier constant:(CGFloat)neg_c;

- (NSLayoutConstraint *)setLayoutWidth:(CGFloat)width;
- (NSLayoutConstraint *)setLayoutHeight:(CGFloat)height;
- (NSLayoutConstraint *)setLayoutWidth:(UIView *)targetView multiplier:(CGFloat)multiplier constant:(CGFloat)c;
- (NSLayoutConstraint *)setLayoutHeight:(UIView *)targetView  multiplier:(CGFloat)multiplier constant:(CGFloat)c;

- (NSLayoutConstraint *)setLayoutCenterX:(UIView *)targetView;
- (NSLayoutConstraint *)setLayoutCenterY:(UIView *)targetView;
- (NSLayoutConstraint *)setLayoutCenterX:(UIView *)targetView constant:(CGFloat)c;
- (NSLayoutConstraint *)setLayoutCenterY:(UIView *)targetView constant:(CGFloat)c;
#pragma mark ----------------------frame适配-----------------------------
/**
 * frame相对父布局适配
 *
 * @note 将指定view及其subview(iphone6下进行frame布局的)的frame参数均乘以适配参数(当前屏幕的宽度和iphone6的宽度比)进行适配
 * @note 适用于布局内控件相对父控件有特定的位置关系且所有控件大小均需适配的场景，比如中间弹窗提示类情景、高度不固定的自定义cell布局等
 * @note 建议storyboard或xib下用autolayout布局完成(按ui设计调好)后，禁用autolayout(移除所有约束)，然后代码中通过执行以下两个接口，确保所有控件的frame都适配好。
 */

/**
 * 相对父布局适配之父视图frame适配
 */
- (void)shiPeiSelf_X_Y_W_H;

/**
 * 相对父布局适配之子视图frame适配
 */
- (void)shiPeiSubView_X_Y_W_H;

@end
```

UIView+CWNView.m

```
#import "UIView+CWNView.h"
#import <objc/runtime.h>

#define SHIPEI(a)  [UIScreen mainScreen].bounds.size.width/375.0*a

@implementation UIView (CWNView)

#pragma mark 布局操作器获取方法

#pragma mark -autolayout布局操作器获取方法

- (void)cwn_makeConstraints:(void (^)(UIView *))block{
    [self setTranslatesAutoresizingMaskIntoConstraints:NO];
    __weak typeof(self) weakSelf = self;
    block(weakSelf);
}

#pragma mark -frame布局适配操作器获取方法

- (void)cwn_makeShiPeis:(void (^)(UIView *))block{
    __weak typeof(self) weakSelf = self;
    block(weakSelf);
}
```

```
#pragma mark ----------------------------------新版本链式编程----------------------------------
#pragma mark ----------------------autolayout布局----------------------------

- (void)setLastConstraint:(NSLayoutConstraint *)lastConstraint{
    objc_setAssociatedObject(self, @selector(lastConstraint), lastConstraint, OBJC_ASSOCIATION_RETAIN_NONATOMIC);
}
- (NSLayoutConstraint *)lastConstraint{
    NSLayoutConstraint *constraint = objc_getAssociatedObject(self, _cmd);
    return constraint;
}

- (UIView *(^)(CGFloat))topToSuper{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(CGFloat) = ^(CGFloat constant){
        [weakSelf setLastConstraint:[weakSelf setLayoutTopFromSuperViewWithConstant:constant]];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)(CGFloat))leftToSuper{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(CGFloat) = ^(CGFloat constant){
        [weakSelf setLastConstraint:[weakSelf setLayoutLeftFromSuperViewWithConstant:constant]];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)(CGFloat))bottomToSuper{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(CGFloat) = ^(CGFloat constant){
        [weakSelf setLastConstraint:[weakSelf setLayoutBottomFromSuperViewWithConstant:constant]];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)(CGFloat))rightToSuper{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(CGFloat) = ^(CGFloat constant){
        [weakSelf setLastConstraint:[weakSelf setLayoutRightFromSuperViewWithConstant:constant]];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)(UIView *, CGFloat, CGFloat))topTo{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(UIView *, CGFloat, CGFloat) = ^(UIView *targetView, CGFloat m, CGFloat c){
        [weakSelf setLastConstraint:[weakSelf setLayoutTop:targetView multiplier:m constant:c]];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)(UIView *, CGFloat, CGFloat))leftTo{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(UIView *, CGFloat, CGFloat) = ^(UIView *targetView, CGFloat m, CGFloat c){
        [weakSelf setLastConstraint:[weakSelf setLayoutLeft:targetView multiplier:m constant:c]];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)(UIView *, CGFloat, CGFloat))bottomTo{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(UIView *, CGFloat, CGFloat) = ^(UIView *targetView, CGFloat m, CGFloat c){
        [weakSelf setLastConstraint:[weakSelf setLayoutBottom:targetView multiplier:m constant:c]];
        return weakSelf;
    };
    return block;
}
```

```
- (UIView *(^)(UIView *, CGFloat, CGFloat))rightTo{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(UIView *, CGFloat, CGFloat) = ^(UIView *targetView, CGFloat m, CGFloat c){
        [weakSelf setLastConstraint:[weakSelf setLayoutRight:targetView multiplier:m constant:c]];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)(CGFloat))width{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(CGFloat) = ^(CGFloat constant){
        [weakSelf setLastConstraint:[weakSelf setLayoutWidth:constant]];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)(CGFloat))height{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(CGFloat) = ^(CGFloat constant){
        [weakSelf setLastConstraint:[weakSelf setLayoutHeight:constant]];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)(UIView *, CGFloat, CGFloat))widthTo{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(UIView *, CGFloat, CGFloat) = ^(UIView *targetView, CGFloat m, CGFloat c){
        [weakSelf setLastConstraint:[weakSelf setLayoutWidth:targetView multiplier:m constant:c]];
        return weakSelf;
    };
    return block;
}
- (UIView *(^)(UIView *, CGFloat, CGFloat))heightTo{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(UIView *, CGFloat, CGFloat) = ^(UIView *targetView, CGFloat m, CGFloat c){
        [weakSelf setLastConstraint:[weakSelf setLayoutHeight:targetView multiplier:m constant:c]];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)(CGFloat))centerXtoSuper{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(CGFloat) = ^(CGFloat constant){
        [weakSelf setLastConstraint:[weakSelf setLayoutCenterX:weakSelf.superview]];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)(CGFloat))centerYtoSuper{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(CGFloat) = ^(CGFloat constant){
        [weakSelf setLastConstraint:[weakSelf setLayoutCenterY:weakSelf.superview]];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)(UIView *, CGFloat))centerXto{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(UIView *,CGFloat) = ^(UIView *targetView, CGFloat c){
        [weakSelf setLastConstraint:[weakSelf setLayoutCenterX:targetView constant:c]];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)(UIView *, CGFloat))centerYto{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)(UIView *, CGFloat) = ^(UIView *targetView, CGFloat c){
        [weakSelf setLastConstraint:[weakSelf setLayoutCenterY:targetView constant:c]];
        return weakSelf;
    };
    return block;
```

```
#pragma mark ----------frame适配----------

- (UIView * (^)())shiPeiSelf{
    __weak typeof(self) weakSelf = self;
    UIView * (^block)() = ^{
        [weakSelf shiPeiSelf_X_Y_W_H];
        return weakSelf;
    };
    return block;
}

- (UIView *(^)())shiPeiSubViews{
    __weak typeof(self) weakSelf = self;
    UIView *(^block)() = ^{
        [weakSelf shiPeiSubView_X_Y_W_H];
        return weakSelf;
    };
    return block;
}
#pragma mark ----------旧版本----------
#pragma mark ----------autolayout布局----------

- (NSLayoutConstraint *)setLayoutTopFromSuperViewWithConstant:(CGFloat)c{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeTop relatedBy:NSLayoutRelationEqual toItem:self.superview attribute:NSLayoutAttributeTop multiplier:1.0f constant:c];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}

- (NSLayoutConstraint *)setLayoutLeftFromSuperViewWithConstant:(CGFloat)c{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeLeft relatedBy:NSLayoutRelationEqual toItem:self.superview attribute:NSLayoutAttributeLeft multiplier:1.0f constant:c];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}

- (NSLayoutConstraint *)setLayoutBottomFromSuperViewWithConstant:(CGFloat)neg_c{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeBottom relatedBy:NSLayoutRelationEqual toItem:self.superview attribute:NSLayoutAttributeBottom multiplier:1.0f constant:-neg
        [self.superview addConstraint:constraint];
    }
    return constraint;
}

- (NSLayoutConstraint *)setLayoutRightFromSuperViewWithConstant:(CGFloat)neg_c{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeRight relatedBy:NSLayoutRelationEqual toItem:self.superview attribute:NSLayoutAttributeRight multiplier:1.0f constant:-neg_c];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}


- (NSLayoutConstraint *)setLayoutTop:(UIView *)targetView multiplier:(CGFloat)multiplier constant:(CGFloat)c{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeTop relatedBy:NSLayoutRelationEqual toItem:targetView attribute:NSLayoutAttributeBottom multiplier:multiplier constant:c];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}

- (NSLayoutConstraint *)setLayoutLeft:(UIView *)targetView multiplier:(CGFloat)multiplier constant:(CGFloat)c{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeLeft relatedBy:NSLayoutRelationEqual toItem:targetView attribute:NSLayoutAttributeRight multiplier:multiplier constant:c];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}

- (NSLayoutConstraint *)setLayoutBottom:(UIView *)targetView multiplier:(CGFloat)multiplier constant:(CGFloat)neg_c{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeBottom relatedBy:NSLayoutRelationEqual toItem:targetView attribute:NSLayoutAttributeTop multiplier:multiplier constant:-neg_c];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}
```

```
- (NSLayoutConstraint *)setLayoutRight:(UIView *)targetView multiplier:(CGFloat)multiplier constant:(CGFloat)neg_c{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeRight relatedBy:NSLayoutRelationEqual toItem:targetView attribute:NSLayoutAttributeLeft multiplier:multiplier constant:-neg_c];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}

- (NSLayoutConstraint *)setLayoutWidth:(CGFloat)width{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeWidth relatedBy:NSLayoutRelationEqual toItem:nil attribute:NSLayoutAttributeNotAnAttribute multiplier:1.0f constant:width];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}

- (NSLayoutConstraint *)setLayoutHeight:(CGFloat)height{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeHeight relatedBy:NSLayoutRelationEqual toItem:nil attribute:NSLayoutAttributeNotAnAttribute multiplier:1.0f constant:height];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}

- (NSLayoutConstraint *)setLayoutWidth:(UIView *)targetView multiplier:(CGFloat)multiplier constant:(CGFloat)c{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeWidth relatedBy:NSLayoutRelationEqual toItem:targetView attribute:NSLayoutAttributeWidth multiplier:multiplier constant:c];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}

- (NSLayoutConstraint *)setLayoutHeight:(UIView *)targetView  multiplier:(CGFloat)multiplier constant:(CGFloat)c{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeHeight relatedBy:NSLayoutRelationEqual toItem:targetView attribute:NSLayoutAttributeHeight multiplier:multiplier constant:c];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}
- (NSLayoutConstraint *)setLayoutCenterX:(UIView *)targetView{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeCenterX relatedBy:NSLayoutRelationEqual toItem:targetView attribute:NSLayoutAttributeCenterX multiplier:1.0f constant:0];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}

- (NSLayoutConstraint *)setLayoutCenterY:(UIView *)targetView{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeCenterY relatedBy:NSLayoutRelationEqual toItem:targetView attribute:NSLayoutAttributeCenterY multiplier:1.0f constant:0];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}

- (NSLayoutConstraint *)setLayoutCenterX:(UIView *)targetView constant:(CGFloat)c{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeCenterX relatedBy:NSLayoutRelationEqual toItem:targetView attribute:NSLayoutAttributeCenterX multiplier:1.0f constant:c];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}

- (NSLayoutConstraint *)setLayoutCenterY:(UIView *)targetView  constant:(CGFloat)c{
    NSLayoutConstraint *constraint;
    if (self.superview != nil) {
        constraint = [NSLayoutConstraint constraintWithItem:self attribute:NSLayoutAttributeCenterY relatedBy:NSLayoutRelationEqual toItem:targetView attribute:NSLayoutAttributeCenterY multiplier:1.0f constant:c];
        [self.superview addConstraint:constraint];
    }
    return constraint;
}
#pragma mark ----------------------frame适配--------------------------

- (void)shiPeiSelf_X_Y_W_H{
    self.frame = CGRectMake(SHIPEI(self.frame.origin.x), SHIPEI(self.frame.origin.y), SHIPEI(self.frame.size.width), SHIPEI(self.frame.size.height));
}

- (void)shiPeiSubView_X_Y_W_H{
    [self.subviews enumerateObjectsUsingBlock:^(__kindof UIView * _Nonnull obj, NSUInteger idx, BOOL * _Nonnull stop) {
        obj.frame = CGRectMake(SHIPEI(obj.frame.origin.x), SHIPEI(obj.frame.origin.y), SHIPEI(obj.frame.size.width), SHIPEI(obj.frame.size.height));
    }];
}
```

(2)　封装库的使用：

```objc
#import "MYHorizontalProgressView.h"
#import "UIView+CWNView.h"

@interface ViewController ()

@property (strong, nonatomic) UIView *redView;
@property (strong, nonatomic) UIView *blueView;

@property (strong, nonatomic) NSLayoutConstraint *redViewTop;//红色视图距离父视图视图顶部约束
@property (strong, nonatomic) NSLayoutConstraint *blueViewHeight;//蓝色视图高度约束

@property (strong, nonatomic) MYHorizontalProgressView *progressView;//可以控制高度的进度条(基于系统控件UIProgressView)

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    __weak typeof(self) weakSelf = self;
    [self.view addSubview:self.redView];
    [self.view addSubview:self.progressView];
    [self.view addSubview:self.blueView];


    //新版本实现
    [_redView cwn_makeConstraints:^(UIView *maker) {
        weakSelf.redViewTop = maker.topToSuper(10).lastConstraint;//记住需动态更新的约束
        maker.leftToSuper(20);
        maker.centerXtoSuper(0);
        maker.bottomTo(weakSelf.progressView, 1, 10);
//      maker.leftToSuper(20).centerXtoSuper(0).bottomTo(weakSelf.progressView, 1, 10);//链式
    }];
   _redViewTop.constant = 20;//动态更新约束


    [self.progressView cwn_makeConstraints:^(UIView *maker) {
        maker.leftToSuper(20).rightToSuper(20).height(10).bottomTo(weakSelf.blueView, 1, 10);
    }];


    //旧版本实现
    [_blueView cwn_makeConstraints:^(UIView *maker) {
        [maker setLayoutLeftFromSuperViewWithConstant:20];
        [maker setLayoutRightFromSuperViewWithConstant:20];
        [maker setLayoutBottomFromSuperViewWithConstant:20];
        weakSelf.blueViewHeight = [maker setLayoutHeight:200];//记住需动态更新的约束
//      maker.leftToSuper(20).rightToSuper(20).bottomToSuper(20).height(200);  链式
    }];

   _blueViewHeight.constant = 150;//动态更新约束



    [self.progressView setProgress:0.5 animated:YES];
}

#pragma mark 控件get方法

- (UIView *)redView{
    if(!_redView){
        _redView = [[UIView alloc] init];
        _redView.backgroundColor = [UIColor redColor];
    }
    return _redView;
}

- (MYHorizontalProgressView *)progressView{
    if(!_progressView){
        _progressView = [[MYHorizontalProgressView alloc] init];
    }
    return _progressView;
}

- (UIView *)blueView{
    if(!_blueView){
        _blueView = [[UIView alloc] init];
        _blueView.backgroundColor = [UIColor blueColor];
    }
    return _blueView;
}
```