

UIKit

2016年5月12日 星期四

上午8:34

UIKit

```
/**
 * 调用self.view.backgroundColor->会调用[self view](getter)方法,
 * 此方法中会创建view对象 即第一次调用对象时才创建对象(延迟加载), 以提高性能
 */
log(@"view: %@", self.view); //第一次调用getter方法时会调用 viewDidLoad 方法
log(@"运用懒加载来创建view对象");
```

```
main(int argc, char *argv[ ], char **env){}
```

argc: 整数,用来统计你运行程序时送给main函数的命令行参数的个数

* argv[]: 字符串数组, 用来存放指向你的字符串参数的指针数组, 每一个元素指向一个参数

1、应用程序启动过程

a.根据第3个参数创建应用程序对象,nil->UIApplication

b.根据第4个参数创建应用程序代理对象delegate

c.启动事件循环

生命周期:

a)didFinishLaunchingWithOptions

b)applicationWillResignActive

c)applicationDidEnterBackground

d)applicationWillEnterForeground

e)applicationDidBecomeActive

f)applicationWillTerminate

IOS应用5种状态


Not Running(非运行状态): 应用没有运行或被系统终止

Inactive(前台非活动状态): 正进入前台, 但不能接受事件处理

Active(前台活动状态): 已经进入前台, 能够处理事件

Background(后台状态): 进入后台依然能够执行代码, 当没有可执行任务后执行完后应用进入

Suspended(挂起状态): 不能执行代码, 若系统内存不够将会被终止



挂起状态

2、UIViewController生命周期

a)viewDidLoad
b)viewWillAppear
c)viewDidAppear
d)viewWillDisappear
e)viewWillDisappear

3、UILabel

作用：显示文本

属性：text, frame, backgroundColor, numberOfLines, font, textColor

每个view都有tag属性 [self.view viewWithTag:100];根据tag属性值找对应的子view

4、UIButton

作用：与用户交互，响应事件

结构：背景层，左侧图片右侧标题

状态：normal, highlighted(按下), selected(btn.selected=YES), disabled(btn.disabled=YES)

设置标题setTitle: forState: 添加事件addTarget: action: forControlEvents:

5、UITextField

属性：text, placeholder ,borderStyle

关闭键盘：让文本框放弃第一响应者身份 [textField resignFirstResponder]

点击view空白处实现touchesBegan方法

点击键盘return键 event->Did End On Exit

弹起键盘：[textField becomeFirstResponder]

leftView -> 密码头像视图, rightView -> 是否可见视图

7、UIStepper 步进控件 & UISlider 滑块控件

属性：value

事件：valueChanged

8、UISwitch 开关控件

属性：on(BOOL)

事件：valueChanged

```
@property(n nonatomic,getter=is0pen)B00L open;  
self.is0pen <=> [self is0pen] <=> self.open
```

9、UIAlertController

AlertView 警告框

1、创建alertView

2、创建意图按钮事件UIAlertAction 并添加addAction到alertView中

3、添加输入框addText... block访问属性：message,textFields

4、推出显示presentViewController

ActionSheet 操作表

10、NSArray self.view.subviews

顺序存放添加进去的view

其他控件

UISegmentedControl(分栏)

selectedSegmentIndex
valueChange

UIActivityIndicatorView(请求菊花)

isAnimating
stopAnimating
startAnimating

UIProgressView(进度条)

progress [0,1]浮点数

UIDatePicker(日期控件)

date

UIPickerView(实现联动控件)

需遵守2个协议 UIPickerViewDataSource, UIPickerViewDelegate

多少列：numberOfComponentsInPickerView

每列多少行：...numberOfRowsInComponent...

每行显示内容: `titleForRow..forComponent...` 选中某行事件: `...didSelectRow...`
`[pickerView reloadData];`//联动刷新
`[self.pickerView selectRow:0 inComponent:1 animated:YES];`//默认选中

11、xib

`@property (weak, nonatomic) IBOutlet UILabel *label;`

xib中拖拽的控件添加到view中, 此时UILabel对象的引用计数已经是1(已经是强引用)

用weak修饰的label变量引用这个对象时, 引用计数不会+1(没必要用strong给它+1)

xib中控件的生命周期 > 属性(weak)label引用的生命周期

12、@class解决两个类的.h文件互相导入问题

用`@class A;`来替换`import "A.h";` 实际上没有导入A的头文件, 只是声明A是一个类型
但如果真要使用这个实例, 还需在B.m中导入`import "A.h"`

前面不能是+号, 如果是+, self表示类本身, 不能调用clickButton

```
8  */
9  -(UIButton*)getButtonWithFrame:(CGRect)rect andTitle:(NSString*)title {
10     [button addTarget:self action:@selector(clickButton:) forControlEvents:UIControlEventTouchUpInside];
11 }
```

13、委托模式IOS(代理java)

1、面向切面编程, 处理共通事务(记录日志, 事务处理, 异常捕获...)

2、委托方 — (实现协议)代理方 双方是可以独立存在的, 互不影响

3、解耦, 修改封闭、扩展开发

自己实现委托

委托方: 制定原则(协议)、声明代理属性、在适当时机调用代理方法

代理方: 遵守原则实现协议、设置委托方代理人为自己

协议中代理声明原则:

方法第一个参数一定是委托方, 尽量体现发送消息的时机

命名: 委托方类名+Delegate

14、响应者链

hit-view 根据触点CGPoint给直接子视图发送hitTest消息，子视图响应(点是否在其中)

事件捕获->事件处理->事件冒泡

(javascript)

Model

数据模型(存储数据)

业务模型(多线程、数据存储、网络处理)

View

Controller

OC中数组、字典、集合等不能放入基本数据类型数据、结构体、枚举

a)NSArray<id> arr = @[.....]; 默认泛型是id类型，只能存放NSObject及其子类

b)基本类型没有内存管理，因为没有继承NSObject(通retain,release,autorelease)来管理存储，
来管理它的内存，会导致内存无法管理(回收)

c)addObject方法会给添加的元素发送retain消息使引用计数count+1，而基本类型数据没有
retain方法，因而拒绝添加到数组中

d)栈中的内存由系统自动管理

泛型

//遍历数组时不能移除元素(不能全部移除)，但反向遍历是可以移除的

```
for (long i = arr.count-1; i >= 0; i--) {  
    [arr removeObjectAtIndex:i];  
}
```

16、界面切换

1、普通切换

```
presentViewController  
dismissViewControllerAnimated
```

2、导航控制器UINavigationController

push

因而不能用引用计数概念

pop

导航栏

高度：64（包含状态栏20像素）

内容：leftBarButtonItem/s title/titleView rightBarButtonItem/s

backgroundImage,backgroundColor,translucent,barStyle,tintColor

toolbar `self.navigationController.toolbarHidden = NO;`

导航->导航 `self.navigationController presentViewController`

当页面只用一次(登录页面)跳转时用 `self.view.window.root...`后面会覆盖，登录页面销毁

17、多VC之间传值: 拿到对方的引用，并且对象要公开能存储数据的属性

正向传值

反向传值：后一页面有前一页面的引用(weak)

18、UIImageView

contentMode属性

ScaleToFill 高宽比例适应frame，图片变形

ScaleAspectFit 比例不变，全部显示，但会留白(缩小)

ScaleAspectFill 比例不变，拉伸部分显示，超出范围(放大)

19、UIScrollView

本身么有外观，主要通过加载子视图来完成内容展示

frame：视图可见空间大小(可见区域)

contentSize：设置内容大小(滚动区域)

contentOffset：左顶点偏移量，移动的是bounds不是内容区域

automaticallyAdjustsScrollViewInsets `//自动内边距调整`

pagingEnabled `//设置整页滑动`

bounces `//关闭弹跳`

scrollView中放标签，CGFloat `offsetX = label.center.x-scrollView.frame.width`

UIPageControl

实现代理UIScrollViewDelegate协议实现滑动页面

numberOfPages,currentPage,pageIndicatorTintColor,
userInteractionEnabled

20、UITableView(继承UIScrollView)

样式: Plain, Group

组成: tableView

section(sectionHeader,UITableViewCell,sectionFooter)
tableViewFooterView

使用: 1、创建实例, 设置frame, 设置数据源(self), 设置代理(self)

2、遵守协议UITableViewDataSource, UITableViewDelegate

3、关注 3 问 1 答(几个分区、行数/分区、内容/row、选中某行处理action)

UITableViewController 自带一个表视图并遵守了2个协议 设置数据源及代理, 只需关注 3 问
cell滑出屏幕会销毁, 新进来的会自动创建, 用缓存技术(空间换时间)

tableView 索引

sectionIndexTitlesForTableView

2种方式:

不用注册, 使用灵活, 但需判断cell能否找到 dequeueReusableCellWithIdentifier

提前注册后取出, 没有系统会自动生成一个返回dequeueReusableCellWithIdentifier

时间复杂度

空间复杂度

UITableViewCell

常用属性 titleLabel,detailTextLabel,imageView

辅助视图 accessoryType

//tableView滚动到最后一行 scrollToRowAtIndex

bounds(大小不变) & frame(旋转缩放会变形)

自定义内容视图cell.contentView

如果cell中没有label, 自定义样式创建一个并添加到cell中, 如果cell中存在label则修

自定义辅助视图cell.accessoryView

动态表格

自定义Block封装数据 放入数组中

1 答

forIndexPath

改其内容、

白底无背景色效果，从八数组中

a)刷新表格：[self.tableView reloadData];

b)更新部分：insertRowsAtIndexPaths

多线程情况下用第二种方式

表格编辑(行添加、删除、移动)

- 1、设置为可编辑
- 2、编辑行的样式
- 3、编辑后响应事件

静态表格：行数不变，设置功能相关界面

下拉刷新

- 1、创建0
- 2、将实例赋给表视图的refreshControl属性
`self.refreshControl = [[UIRefreshControl alloc] init];`
- 3、给refresh添加valueChange事件 addTarget

集合视图控制器

numberOfSectionsInCollectionView (多少分区)

numberOfItemsInSection (没分区多少项)

cellForItemAtIndexPath (自定义创建UICollectionViewCell)

`cell.displayLabel.text`

手动设置：UICollectionViewDataSource

创建UICollectionViewFlowLayout 设置布局 (flowLayout)

`[[UICollectionView alloc] initWithFrame:frame collectionViewLayout:flowLayout];`

`collectionView.dataSource = self` 设置代理

绑定视图控制器，绑定布局，重写cell的initWithFrame，重写layout的initWithCoder

shouldInvalidateLayoutForBoundsChange 当边界改变时重新计算布局

获取每一个Cell的布局属性，通过attributes.transform3D实现Cell的缩放动画

layoutAttributesForElementsInRect

owLayout]

选项卡控制器

tabBar的下标依次对应tab

```
bar.tabBar.backgroundColor
```

```
bar.viewControllers
```

```
view.tabBarItem.title/badgeValue/image/selectedImage
```

UISearchDisplayController

TableView中的搜索 <UISearchDisplayDelegate>

实现方法，在该方法中处理搜索结果，返回yes自动reload表视图

```
searchDisplayController:shouldReloadTableForSearchString:
```

```
searchDisplayController:shouldReloadTableForSearchScope:
```

UISearchDisplayController/UISearchBar

NSTimer (单线程, 循环队列)

```
timerWithTimeInterval
```

需手动添加到循环队列中

```
[[NSRunLoop currentRunLoop]addTimer:_timer forMode:NSDefaultRunLoopM
```

```
scheduledTimerWithTimeInterval
```

fire 触发

```
invalidate 使之失效
```

//new/alloc/copy 单例创建失败 推荐使用GCD(多线程安全)

1、static -> private static(java)只能是本文件内访问

2、单例模式 放方法中(限制作用域)只能在此方法中访问

3、全局变量在程序中任何地方都可访问, 其他文件中访问时需声明extern(表明变量是外部引

打印基本类型数据

NSStringFromCGxxx

ode]

进的全局变量)

Storyboard

1、将很多xib集中到一个文件中，特点是用场景(Scene)代替xib文件。

负责创建和管理所有界面(VC,Scene)间的跳转，以提高开发效率。

通过一个标识 找到 segue(vc之间的连线) 指向的那个控制器并跳转

```
[self performSegueWithIdentifier:@"id" sender:nil];
```

通过标识取出一个场景，返回场景绑定的控制器

```
[self.storyboard instantiateViewControllerWithIdentifier:@"login"]
```

通过xib文件加载一个视图控制器

```
[self initWithNibName:@"News" bundle:nil]
```

根据storyboard的名字实例化对象，并自动创建默认的视图控制器

```
[[UINavigationController storyboardWithName:@"MySB" bundle:nil] instantiateInitialViewController]
```

定义全局导航样式

```
+(void)initialize{
```

```
//获取导航控制器中导航条的外观
```

```
UINavigationController *bar = [UINavigationController appearance];
```

```
    自定义样式
```

```
}
```

定义全局TabBar

```
+(void)initialize{
```

```
    if (self == [TarBar class]) {
```

```
        UITabBar *bar = [UITabBar appearance];
```

```
        //获取 UITabBarItem 的样式
```

```
        UITabBarItem *item = [UITabBarItem appearance];
```

```
        // 自定义样式
```

```
    }
```

```
}
```

获取导航栏和状态栏的高度

```
CGRect rect = [[UIApplication sharedApplication] statusBarFrame];
```

设置TabBarItem图片为原色，先取出图片设置为不跟随intColor而改变，再设置Item图片

```
[image imageWithRenderingMode:UIImageRenderingModeAlwaysOriginal]
```

intColor

tialViewController]

设置父视图tintColor，则所有未单独设置过tintColor的子视图都继承父视图的tintColor

9切片：使用代码或在storyboard中直接设置，将图片分为9块，在图片拉伸形变时四角内容不

设置icon：对应尺寸图片放到AppIcon中对应位置

屏幕尺寸图片适配：5s (320*568) 实际尺寸(640*1136) 2x

image.png(4) image@2x.png(4s,5,5s,6,6s) image@3x.png(6+,6s+)

位图(点阵图)：用RGB保存，4字节保存一个点 缩放会失真

矢量图：保存生成图形的公式和函数等，缩放时会根据公式自动计算需要多少个点

IOS本身不识别jpg，默认将jpg->png渲染

渲染引擎：只识别 2^n ，128*128->选128*128渲染，129*129->选256*256渲染

OC对象与图形转让

Core Graphics(核心绘图) 一套C语言函数库，即Quart 2D苹果设备(IOS,OSX)的2D绘图引擎

绘制图形

a)在视图类(UiView及其子类)中重写drawRect方法，由系统调用

准备画板、勾勒图形、调色、绘制(填充和描边)、保存

b)UIBezierPath(直线、曲线、圆形、扇形、多边形...)

1、获取上下文 CGContextRef context = UIGraphicsGetCurrentContext()

2、获取贝塞尔曲线路径对象 UIBezierPath *path = [UIBezierPath bezierPath]

3、勾勒图形 (addLineToPoint 设置起点，终点，封闭路径closePath)

4、设置样式 (配置颜色[color setStroke],线宽lineWidth,lineJoinStyle,lineCapStyle)

5、保存路径 CGContextSaveGState(context)

绘制弧形： bezierPathWithArcCenter : radius : startAngle : endAngle : clockwise

绘制矩形： bezierPathWithRoundedRect

绘制椭圆： bezierPathWithOvalInRect

绘制字符串： [UIBezierPath bezierPathWithText:string inRect:context:font:baseline:align:]

变

le, 填充和描边)

绘制字符串: `str drawInRect:CGRect withAttributes:dict`

绘制曲线: `addCurveToPoint :point :controlPoint1 :controlPoint2`

绘制字符串

1、自适应高度和宽度

a) `CGRect frame = [str boundingRectWithSize:CGSizeMake(WIDTH, HEIGHT) options:NSStringDrawingUsesLineFragmentOrigin attributes:dict context:]`
b) `[str drawInRect:CGRectMake(50, 100, frame.size.width, frame.size.height) withAttributes:dict]`

2、动态获取label高度

`sizeThatFits` 根据给定size返回最合适的大小

创建临时画布 `UIGraphicsBeginImageContext(size)`

在画布中画东西 在封闭区域内显示内容, 其他区域无效 `[path addClip]`

把图片对象画在某个区域内 `[image drawInRect:self.view.bounds]`

从画布中生成图片对象 `UIGraphicsGetImageFromCurrentImageContext()`

二维码

// 创建一个二维码种类路径

`CIFilter *filter = [CIFilter filterWithName:@"CIQRCodeGenerator"];`

`CIImage128BarcodeGenerator` 条形码

// 恢复滤镜为默认设置

`[filter setDefaults];`

// 准备元数据

`NSString *baidu = @"http://www.baidu.com";`

`NSData *data = [baidu dataUsingEncoding:NSUTF8StringEncoding];`

// 设置 滤镜 `inputMessage` 属性

`[filter setValue:data forKey:@"inputMessage"];`

// 输出二维码图片

`CIImage *image = [filter outputImage];`

// 设置为imageView的图片

`self.imageView.image = [UIImage imageWithCIImage:image];`

```
)  
t:nil]  
height)
```


UITouch

touchesBegan

```
[self.bezierPath moveToPoint:[[touches anyObject] locationInView:self.view]]
```

touchesEnded

```
[self.bezierPath addLineToPoint:[[touches anyObject] locationInView:self.view]]
```

```
[self setNeedsDisplay]//重复绘制
```

touchesMoved

手势

步骤：创建手势对象，设置属性，将手势添加到视图中

重复添加手势事件只有最后的生效

点击 UITapGestureRecognizer

长按 UILongGestureRecognizer

轻扫 UISwipeGestureRecognizer

缩放 UIPinchGestureRecognizer

拖拽 UIPanGestureRecognizer

旋转 UIRotationGestureRecognizer

核心属性

state 手势状态

view 手势中的视图

numberOfTapsRequired 点击次数

numberOfTouchesRequired 多点触摸

direction 轻扫方向

minimumPressDuration 长按时间

rotation 旋转弧度

velocity 缩放速率 scale 缩放比例

位置 locationInView

总偏移量(同步移动时注意清零) translationInView

transform 视图变形

常量CGAffineTransformIdentity记录矩阵没有变化的初始6个值(对象线3个1 其它0) 可用

```
f]]//起始点
```

```
self]]//画线
```

干清除视图transform属性

带Make的函数基于原始位置变化，不带Make的函数基于上一次变化的位置变化

带Make的函数基于原始位置变化，不带Make的函数基于上一次变化的位置变化

旋转 `CGAffineTransformMakeRotation(rotation)/CGAffineTransformRotate`

缩放 `CGAffineTransformScale(imageView.transform, scale, scale)/CGAffineTransformScale`

位移 `CGAffineTransformMakeTranslation/CGAffineTransformTranslate`

多个手势共存需 `UIGestureRecognizerDelegate`，并设置手势对象的代理为当前控制器

坐标系4成员

frame: 描述此视图在父视图中位置和占父视图空间的大小(在父视图中的定位)

bounds: 描述的是视图自身的大小，其值和父视图无关，改变会影响子视图

origin→none size→frame

transform: 描述一个视图的变形(缩放、旋转、位移)，不能和自动布局同时使用

位移: frame原点变, bounds不变, center不变

旋转: frame原点及大小都变, bounds不变, center不变

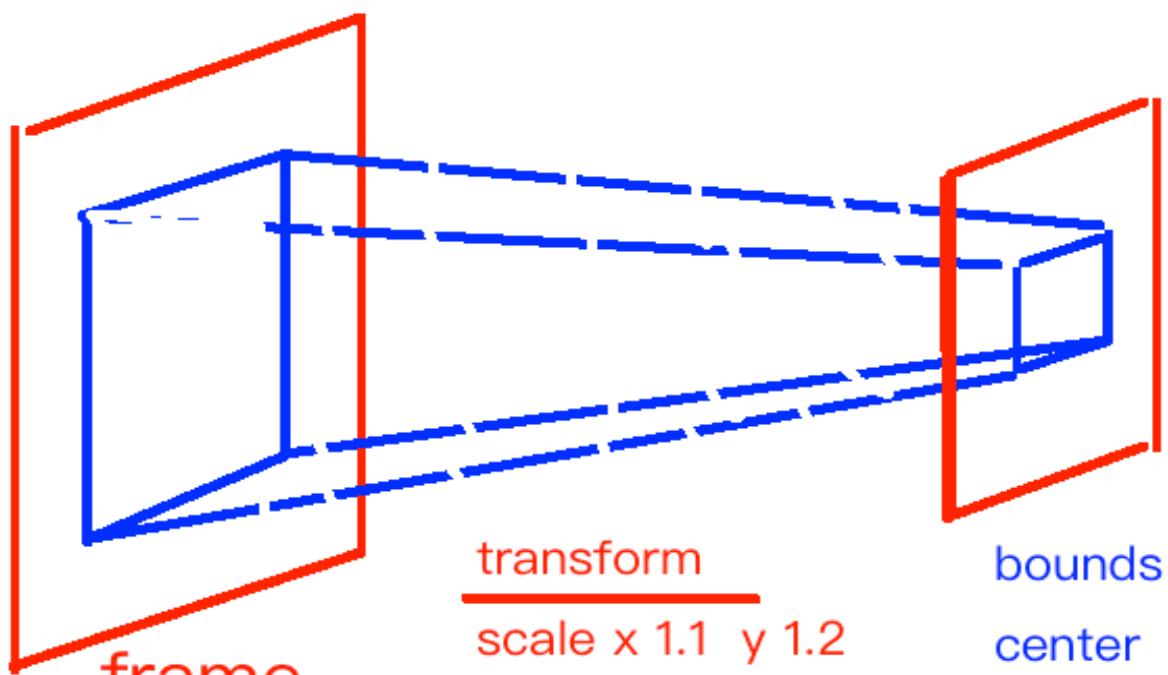
缩放: frame原点及大小都变, bounds不变, center不变

center: 视图中心

-(id)initWithCoder 故事版创建时调用

设备控件 (看见)

用户控件 (看不见)



neTransformMakeScale

映射规则

UIImagePickerController 打开相机/相册

picker.sourceType = 相机/相册

picker.allowsEditing = YES 允许对选中的图片编辑

UINavigationControllerDelegate, UIImagePickerControllerDelegate

实现代理UIImagePickerController: didFinishPickingMediaWithInfo:

/** 合成图片并保存到相册 */

```
- (IBAction)buttonClick:(id)sender {
    UIImage *image = [self getImageFromView:self.mainImage];
    // 保存到相册
    UIImageWriteToSavedPhotosAlbum(image, self, @selector(image:didFinishSavingWithError:co
}
```

/** 从当前view中生成image */

```
-(UIImage *)getImageFromView:(UIView *)view{
    // 创建画布
    UIGraphicsBeginImageContext(view.frame.size);
    // 将view内容渲染到画布中
    [view.layer renderInContext:UIGraphicsGetCurrentContext()];
    // 从画布中取出图片
    UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return image;
}
```

动画

1、帧动画 (每一帧静态图片连续播放组成动画)

[UIImage animatedImageNamed:@"ship" duration:1/10] //必须是png图片最多10

[UIImage animatedImageWithImages:images duration:1/12]

UIImageView动画属性

animationImages

animationDuration

animationRepeatCount 0表示无限重复

2、NSTimer

NSRunLoop

3、UIView

animateWithDuration

ntextInfo:), nil);

张

4、转场动画

`transitionFromView` 从一个view到另一个view的转场动画

`transitionWithView` 这个view以什么形式出现,options使用带Transition的参数

5、CALayer

UIView动画依赖于CALayer实现, CALayer不能响应事件(UIResponse)

`borderWidth/Color masksToBounds`遮罩外面不显示(圆头像)

`cornerRadius、position、阴影`

具有层级性: 自定义layer添加到另一个layer中

`CALayer`

`CATextLayer` 字符串layer

`CAShapeLayer` 图形layer

绘制圆形图片

```
self.layer.borderColor = [UIColor greenColor].CGColor;
self.layer.borderWidth = 3;
self.layer.cornerRadius = self.bounds.size.width * 0.5;
self.layer.masksToBounds = YES;
self.layer.anchorPoint = CGPointMake(0.5, 0.5);
```

6、高级动画CAAnimation

`CABasicAnimation` (位移、旋转、缩放、变形)

`CAKeyframeAnimation` 关键帧动画

`CAAnimationGroup` 动画组

属性: `keyPath` (可以单独设置transform.rotation,position/position.y,transfo

`fromValue、toValue、duration、repeatCount、fillMode`(动画结束时状态)

`removedOnCompletion、beginTime、timingFunction`

断言: `assert(a == b)`; 若 `a != b`, 程序报错

布局

1、**代码布局**: 当屏幕发生变化自动重新计算各个视图的frame(重新定位)

`viewDidLayoutSubviews` 重写, 视图从无到有、旋转都会自动调用此方法

`self.topLayoutGuide.length`

`self.bottomLayoutGuide.length`

rm.scale)

2、Autoresizing

3、Auto Layout

自动布局(约束): 通过一系列的约束constraint来描述视图展示位置

注意: 1 & 2 可以混用, 但要关闭3(AutoLayout)

`translatesAutoresizingMaskIntoConstraints` 自动布局时要关闭Autoresizing

全屏幕适配与SizeClasses配合使用

手动修改约束: 关联 `NSLayoutConstraint` 到父视图控制器中

万能公式: $\text{view1.attr} <\text{relation}> \text{view2.attr} * \text{multiplier} + \text{constraint}$

VFL可视化格式化语言:

|: 父视图边缘

H: 水平方向

V: 垂直方向

[]: 一个子视图(控件)

(): 高 宽 的条件

–: 标准间距 8

–x–: 间距 x

eg: |–20–button1–10–button2–10–button3–20–|

`layoutIfNeeded` 需要时重新布局

`layoutSubviews` 创建、屏幕旋转、点击cell时调用

`updateConstraints` 创建时调用

对约束做动画, 需要在UIView做动画的方法外面设置约束值, 在里面调用`layoutIfNeeded`

键盘通知, 改变底部约束

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:  
@selector(openKeyboard:) name:UIKeyboardWillShowNotification object:
```

通知

完成对象之间的消息传递, 观察者模式

重新布局

nil];

推送 -> 通知中心 <- 订阅

单例对象

系统单例：UIApplication、NSNotificationCenter、NSUserDefaults、NSFileManager、NSU

推送：postNotification...

移除：removeObserver...(对象销毁前dealloc中需移除该对象的所有通知)

```
AppDelegate *delegate = [UIApplication sharedApplication].delegate
[UIApplication sharedApplication].windows.lastObject -> self.view
```

```
- (UIStatusBarStyle)preferredStatusBarStyle {
    return UIStatusBarStyleLightContent;
}
- (BOOL)prefersStatusBarHidden {
    return YES;
}
```

hittest响应者链

1、事件捕获阶段

后加入的视图一定是在先加入视图的上层，先判断最后加入的视图(用户交互、是否隐藏、

2、事件处理阶段

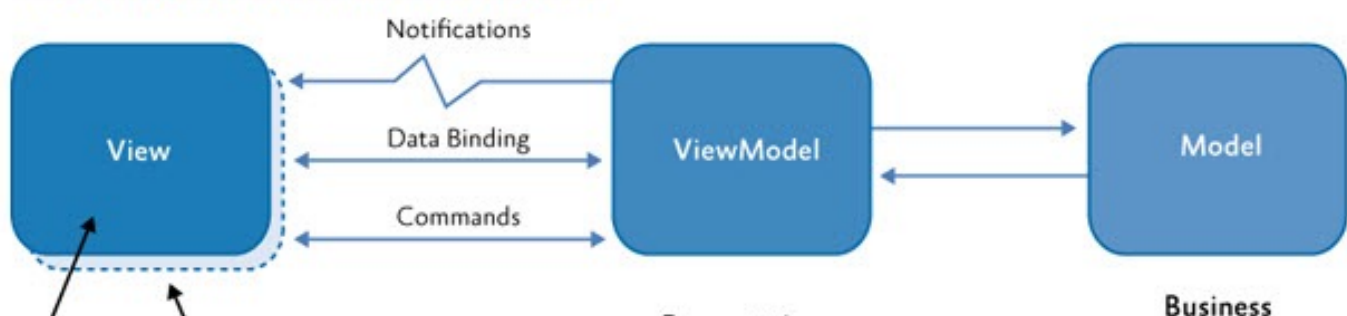
3、事件冒泡阶段

MVVM 设计模式

<http://www.cocoachina.com/ios/20150526/11930.html>

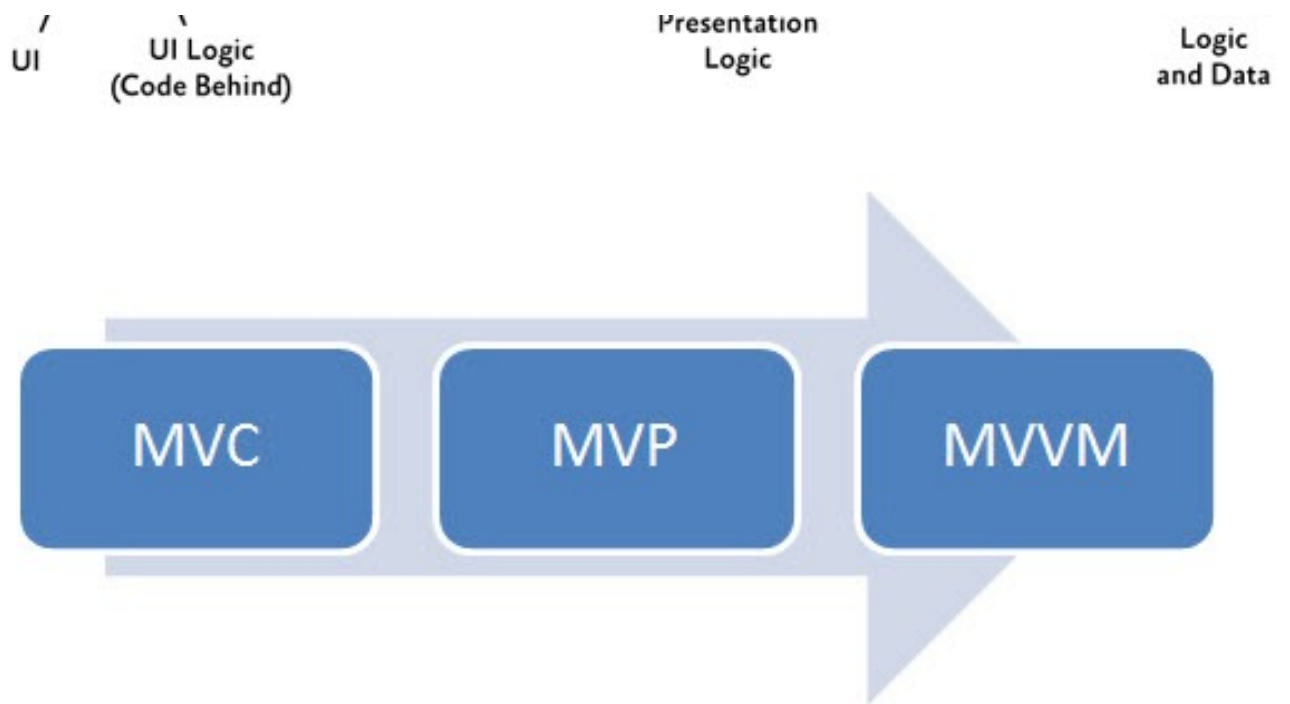
<http://blog.leichunfeng.com/blog/2016/02/27/mvvm-with-reactivecocoa/>

The MVVM classes and their interactions



RLCache

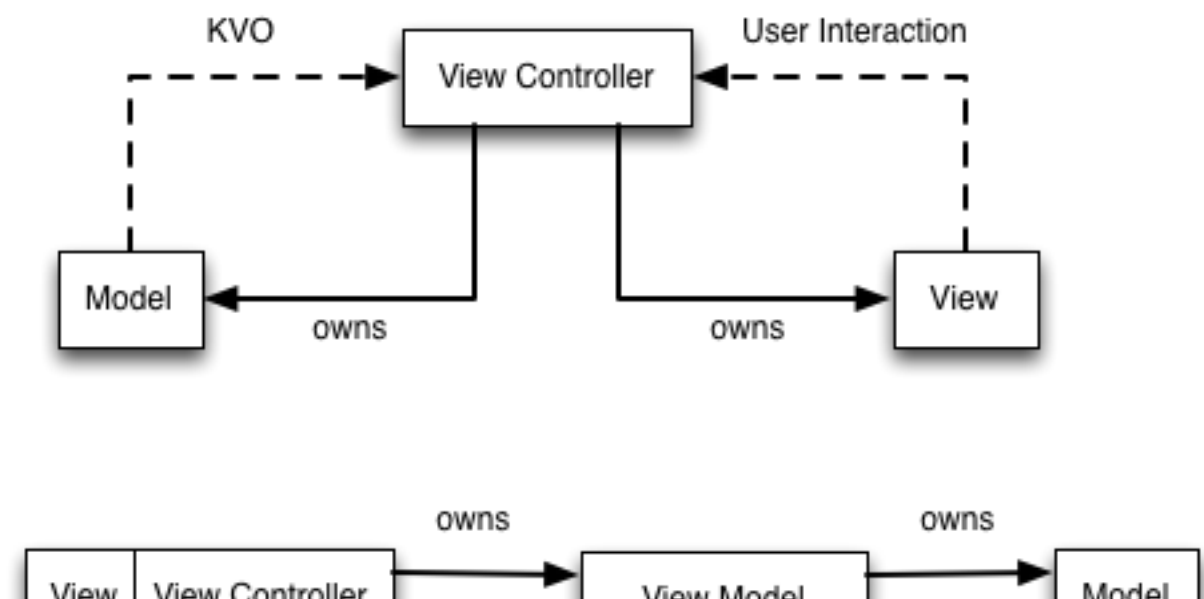
透明度、point)



广泛应用于WPF项目开发中，使UI和业务逻辑分离，让UI设计人员和业务逻辑人员分工明确
WPF重要概念：模板、依赖属性、数据绑定

- 1、View中不可写业务逻辑
- 2、ViewModel层不是Model层的简单封装，也不是View层的简单映射
- 3、ViewModel主要负责显示逻辑、验证逻辑、网络请求等，Model的表现逻辑通过它暴露给View

IOS--MVC



合View



MVVM兼容MVC，将View逻辑放到ViewModel中，带来更轻量级的ViewController配合绑定机制（ReactiveCocoa、KVO），同步更新数据

函数响应式编程ReactiveCocoa MVVM 具有低耦合、可重用性、独立开发、可测试性等优势。FRP提供了一种信号机制来实现这样的效果，通过信号来记录值的变化。信号可以被叠加、分发。通过对信号的组合，就不需要去监听某个值或事件。提供一个单一的、统一的方法去处理异步行为。delegate, block, target-action, notification, KVO

Signal and Subscriber（RAC核心内容）

Signal获取数据后，会调用Subscriber的sendNext, sendComplete, sendError来传送数据。

eg: [signal subscribeNext: error: completed:] 只要没有sendComplete和sendError，sendNext就会一直传值

```

[RACObserve ( self , username ) subscribeNext : ^ ( NSString* newName )
    NSLog ( @"newName : %@", newName ) ;
}];
  
```

RACObserve通过KVO监听property变化，当值被自己或外部改变，block就会执行。修改（map）、过滤（filter）、叠加（combine）、串联（chain）

点
分割或合并。通
行为，

居
ndNext会不断