# 每天一个问题

## 1、[super init]的作用

## 2、画圆，画圆角4+1（CC）=5中方式

### 第一种：使用drawRect方法绘图

自定义一个**View**，在其**drawRect**方法中操作，尽量避免使用**drawRect**方法，会耗性能，资源浪费

- (**void**)drawRect:(CGRect)rect {

//一个不透明类型的Quartz 2D绘画环境，相当于一个画布,你可以在上面任意绘画

//获取绘图上下文

CGContextRef context = UIGraphicsGetCurrentContext();

设置各种字体颜色 格式

最后用CGContextDrawPath方法，画在画布上

CGContextDrawImage画图片

}

### 第二种：设置secondView.layer.cornerRadius = 50;

### secondView.layer.masksToBounds = YES;

会引起离屏渲染

### 第三种：使用贝塞尔曲线UIBezierPath和Core Graphics框架在图片的绘图上下文中画出一个透明的圆。

增加CPU负担，增加内存开销

UIImageView \*imageViewTwo = [[UIImageView alloc]initWithFrame:CGRectMake(100, 100, 100, 100)];

imageViewTwo.image = [UIImage imageNamed:@"imageName1.jpg"];

//开始对imageView进行画图，在透明的这个imageView的画布上开始画

UIGraphicsBeginImageContextWithOptions(imageViewTwo.bounds.size, NO, [UIScreen mainScreen].scale);

// UIGraphicsBeginImageContextWithOptions(<#CGSize size#>, <#BOOL opaque#>, <#CGFloat scale#>)

//opaque：不透明

//使用贝塞尔曲线画出一个圆形图，并且切掉其他的，就剩下了透明的想要的那个圆

[[UIBezierPath bezierPathWithRoundedRect:imageViewTwo.bounds cornerRadius:imageViewTwo.frame.size.width] addClip];

// + (instancetype)bezierPathWithRoundedRect:(CGRect)rect cornerRadius:(CGFloat)cornerRadius;

//rounds all corners with the same horizontal and vertical radius

[imageViewTwo drawRect:imageViewTwo.bounds];

imageViewTwo.image = UIGraphicsGetImageFromCurrentImageContext();

//结束画图

UIGraphicsEndImageContext();

[self.view addSubview:imageViewTwo];

### 第四种：使用CAShapeLayer和UIBezierPath设置圆角，设置view.layer.mask

同样会触发离屏渲染。

UIImageView \*imageView = [[UIImageView alloc]initWithFrame:CGRectMake(100, 100, 100, 100)];

imageView.image = [UIImage imageNamed:@"imageName1.jpg"];

UIBezierPath \*maskPath = [UIBezierPath bezierPathWithRoundedRect:imageView.bounds byRoundingCorners:UIRectCornerAllCorners cornerRadii:imageView.bounds.size];

CAShapeLayer \*maskLayer = [[CAShapeLayer alloc]init];

//设置大小

maskLayer.frame = imageView.bounds;

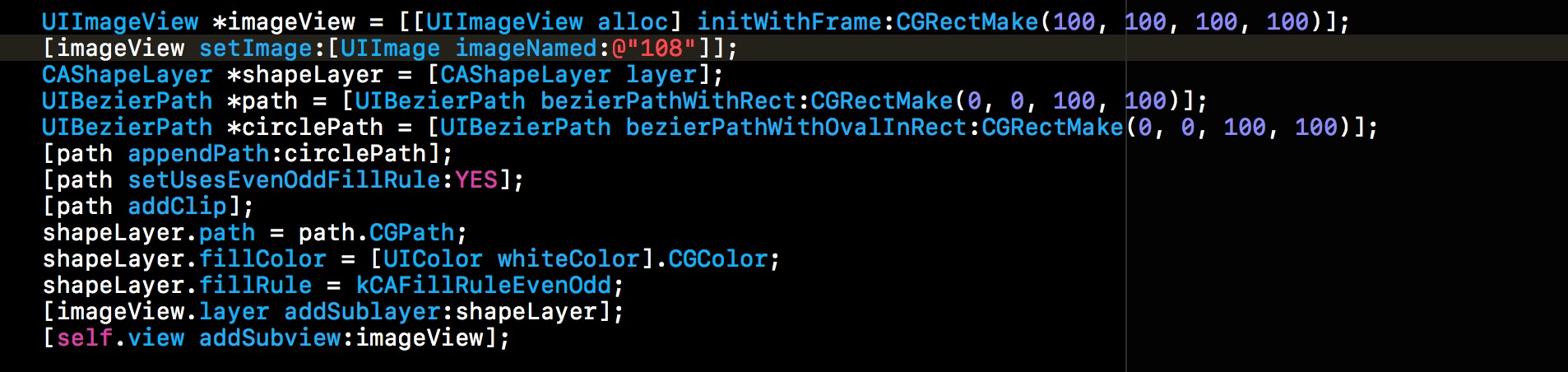
//设置图形样子

maskLayer.path = maskPath.CGPath;

imageView.layer.mask = maskLayer;

[self.view addSubview:imageView];

### 第五种：CC的方法，addsubLayer



### 第六种：混合两个imageView，遮盖下面的那个，形成圆角

需要切图配合，更直接的，就直接让切图给一个圆角图就可以了。

总结：

尽量避免重写 drawRect 方法。不恰当的使用这个方法会导致内存暴增。举个例子，iPhone6 上与屏幕等大的 UIView，即使重写一个空的 drawRect 方法，它也至少占用 750 \* 1134 \* 4 字节 ≈ 3.4 Mb 的内存。在 [内存恶鬼drawRect](http://bihongbo.com/2016/01/03/memoryGhostdrawRect/) 及其后续中，作者详细介绍了其中原理，据他测试，在 iPhone6 上空的、与屏幕等大的视图重写 drawRect 方法会消耗 5.2 Mb 内存。总之，能避免重写 drawRect 方法就尽可能避免。

最基础的一种UIView.layer.cornerRadius

maskToBounse =yes

设置 masksToBounds 会导致离屏渲染，出现性能降低

label 的四周出现了黄色的标记，说明这里出现了**离屏渲染**

高效设置圆角

为普通的 UIView设置圆角，和为 UIImageView 设置圆角的原理截然不同。

4中画圆角的方式

离屏渲染：

指的是GPU在当前屏幕缓冲区以外新开辟一个缓冲区进行渲染操作。

iOS 9.0 之前UIimageView跟UIButton设置圆角都会触发离屏渲染

iOS 9.0 之后UIButton设置圆角会触发离屏渲染，而UIImageView里png图片设置圆角不会触发离屏渲染了，如果设置其他阴影效果之类的还是会触发离屏渲染的。

## 3、tabBar左右滑动的问题

## 4、tableView加timer

## 5、客户详情的右上角的聚合按钮

## 6、keyWindow的问题

## 8、mjrefresh实现

## 9、ASI AFN

## 10、srwebsocket 内部实现 保活机制

## 11、keywindow

## 12、界面等比放大

## 13、ARC 下 viewdidload 中定义的变量什么时候会释放

## 14、数据库升级

## 15、uicontrol uiresponse

## 16、block 为啥不能修改局部变量

## 17、内存管理 strong weak

## 18、自动释放池 三个，加到哪一个

## 19、KVC KVO

## 20、栈和堆

## 21、static

## 22、http 和 https

## 23、iOS目录结构

## 24、framework .a 静态库和动态库

## 25、socket

## 26、bitcode

## 27、死锁产生的条件，怎么避免

四个必要条件：互斥条件    请求与保持   不剥夺条件    循环等待条件

## 28、tcp udp

## 29、userdefault 存储的数据能通过物理路径找到吗

## 30、获取帧率

## 31、深入理解RunLoop

## 32、深入理解GCD