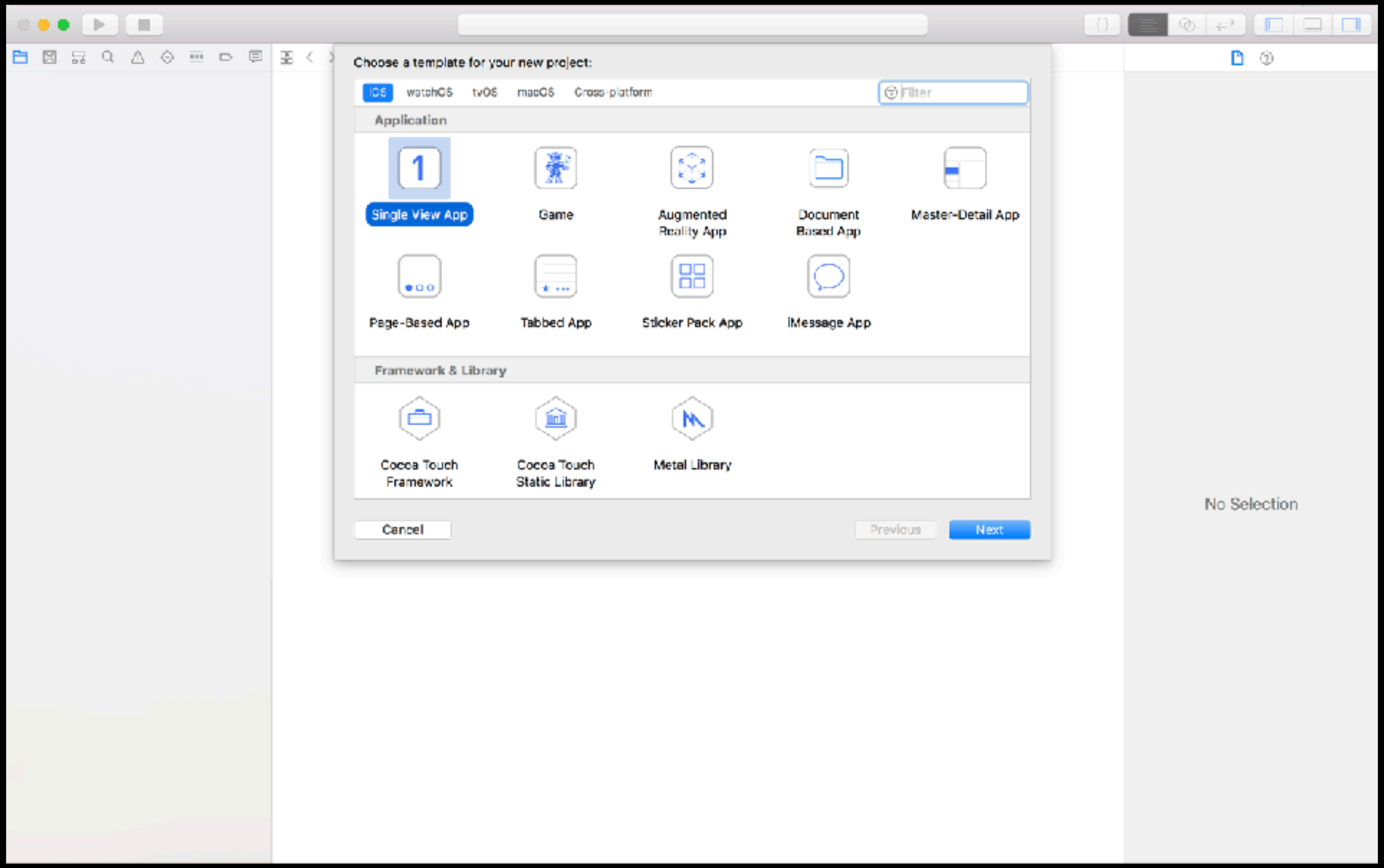
iOS UI & UIKit

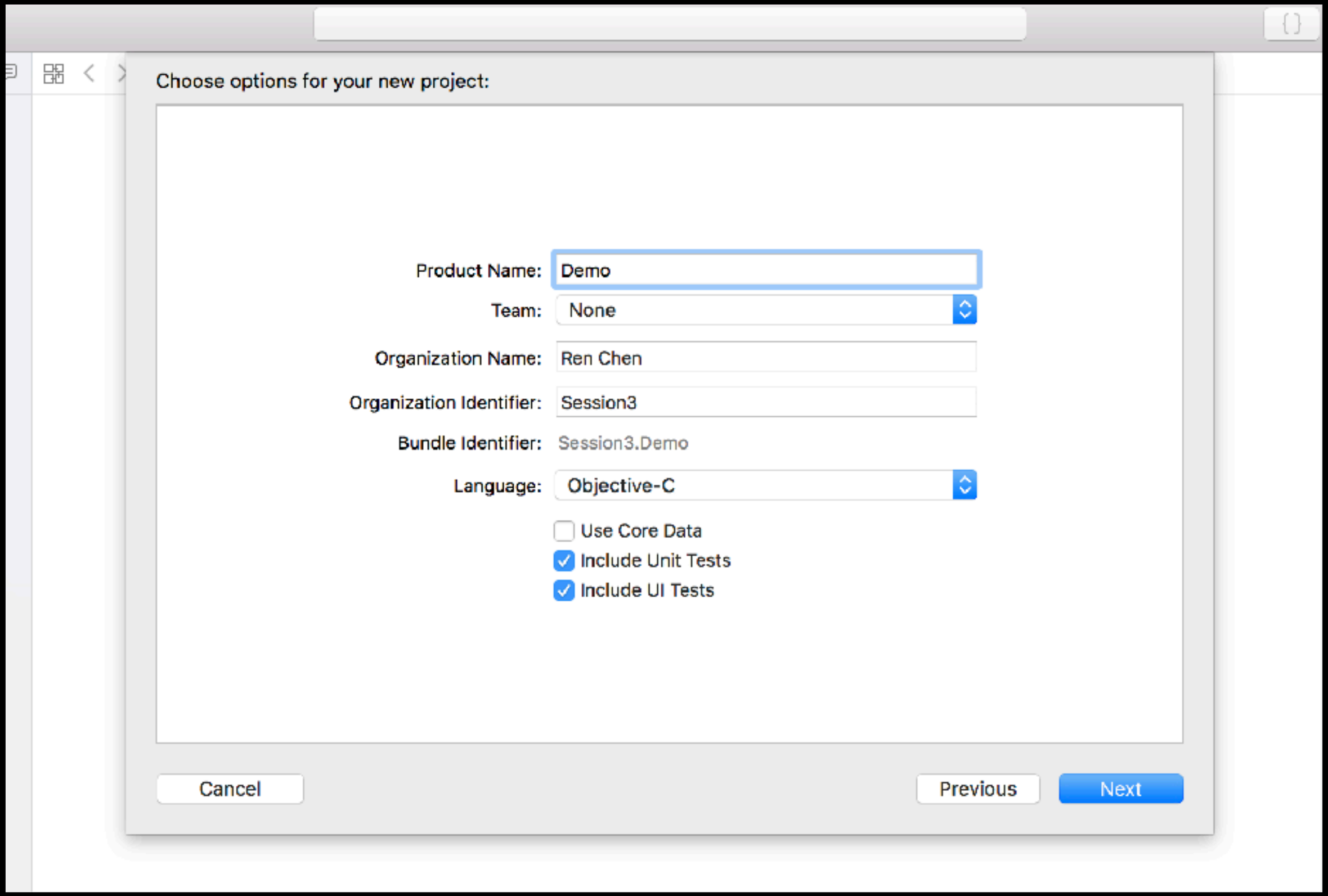
Session 3　开始创建一个简单的app

陈壬　抖音iOS研发工程师

创建一个简单的iOS App



创建一个简单的iOS App



创建一个简单的iOS App

ByteDance 字节跳动

ImagesViewDemo
- ImagesViewDemo
  - s0.jpg
  - s1.jpg
  - s2.jpg
  - s3.jpg
  - s4.jpg
  - s5.jpg
  - s6.jpg
  - s7.jpg
  - s8.jpg
  - s9.jpg
  - AppDelegate.h
  - AppDelegate.m
  - ViewController.h
  - ViewController.m
  - Main.storyboard
  - Assets.xcassets
  - LaunchScreen.storyboard
  - Info.plist
  - main.m
- ImagesViewDemoTests
  - ImagesViewDemoTests.m
  - Info.plist
- ImagesViewDemoUITests
  - ImagesViewDemoUITests.m
  - Info.plist
- Products

General | Capabilities | Resource Tags | Info | Build Settings | Build Phases | Build Rules

PROJECT
- ImagesViewDemo

TARGETS
- ImagesViewDemo
- ImagesViewDemoTe...
- ImagesViewDemoUI...

Identity

Display Name: ImagesViewDemo
Bundle Identifier: Session3.ImagesViewDemo
Version: 1.0
Build: 1

Signing

Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

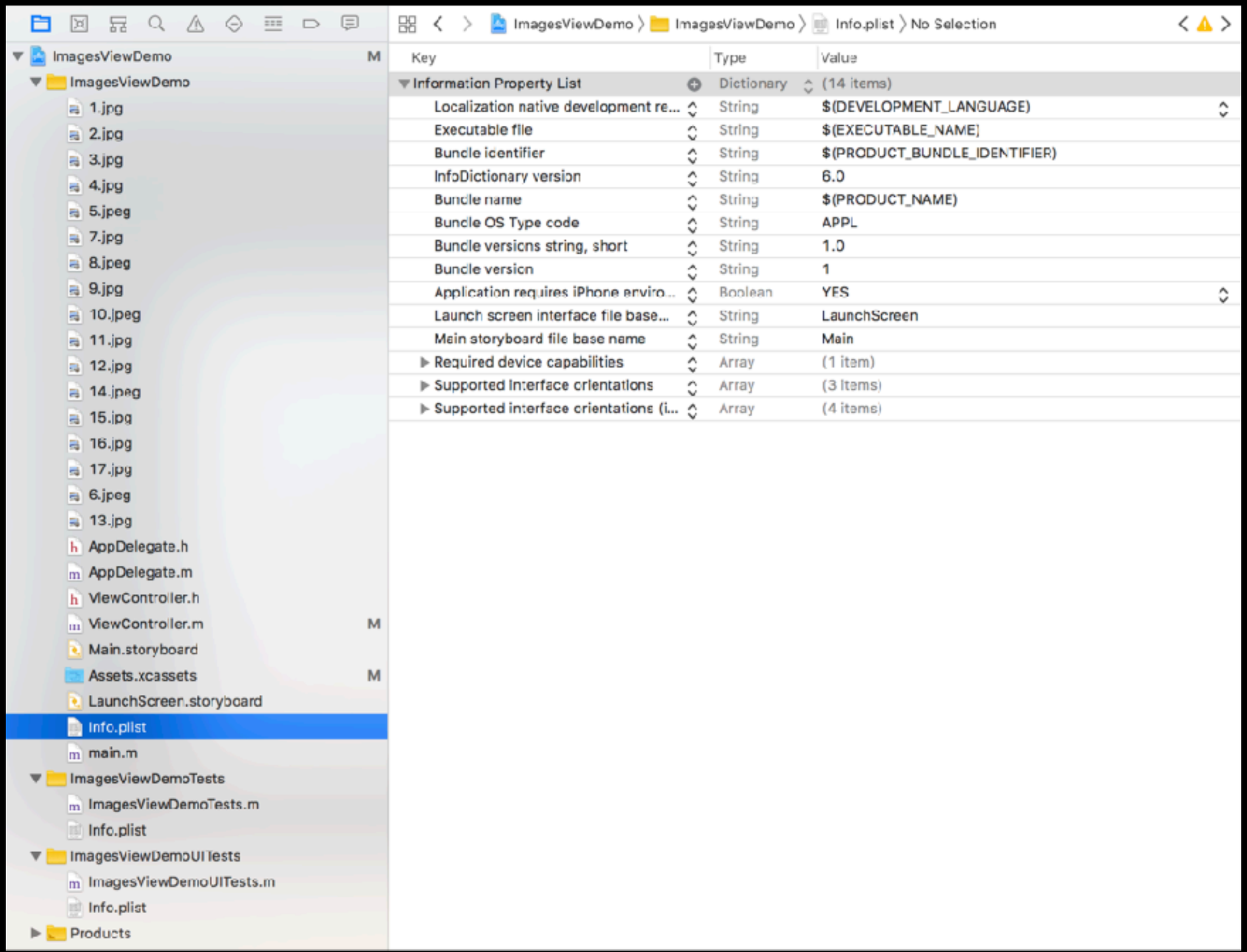
Team: None
Provisioning Profile: Xcode Managed Profile
Signing Certificate: iOS Developer

Status: Signing for "ImagesViewDemo" requires a development team.
Select a development team in the project editor.

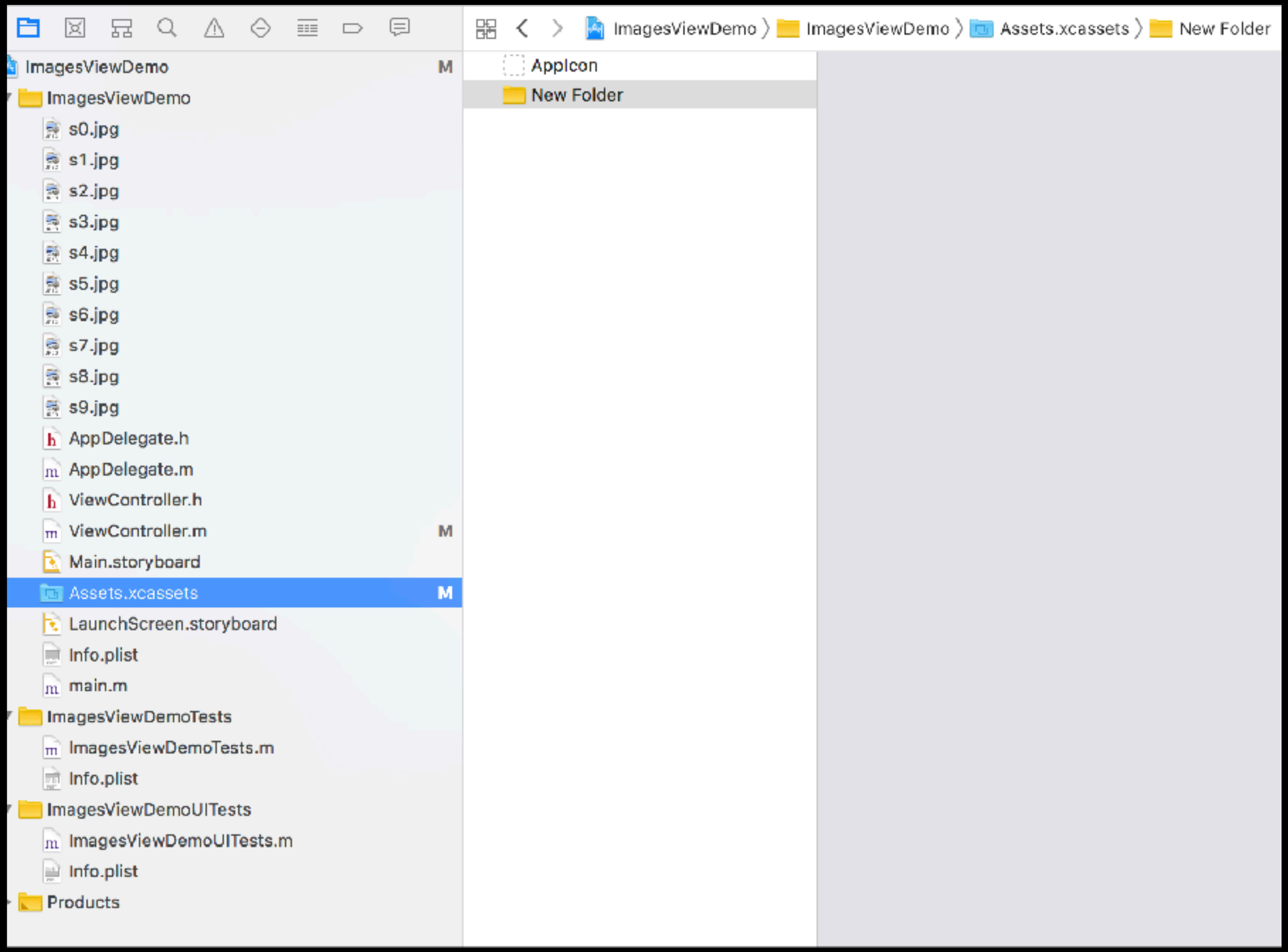
Deployment Info

Deployment Target: 12.1
Devices: Universal
Main Interface: Main
Device Orientation: Portrait, Upside Down, Landscape Left, Landscape Right
Status Bar Style: Default
Hide status bar
Requires full screen

创建一个简单的iOS App - Info.plist简介



创建一个简单的iOS App - Info.plist简介



创建一个简单的iOS App

ByteDance 字节跳动

ImagesViewDemo 〉 iPhone XS

Running ImagesViewDemo on iPhone XS

ImagesViewDemo PID 28717

CPU	0%
Memory	45.7 MB
Disk	480 KB/s
Network	Zero KB/s

- Thread 1 Queue: com.apple.main-thread (serial)
  - 0 -[AppDelegate application:didFinishLaunchingWithOptions:]
  - 1 -[UIApplication _handleDelegateCallbacksWithOptions:isSus...
  - 29 UIApplicationMain
  - 30 main
  - 31 start
  - 32 start
- Thread 2
  - 0 __workq_kernreturn
  - 2 start_wqthread
- Thread 3
  - 0 __workq_kernreturn
- Thread 4
  - 0 __workq_kernreturn
  - 2 start_wqthread
- Thread 5
  - 0 __workq_kernreturn
  - 2 start_wqthread
- com.apple.uikit.eventfetch-thread (6)

ImagesViewDemo 〉 ImagesViewDemo 〉 main.m 〉 main

```
1  //
2  //  main.m
3  //  ImagesViewDemo
4  //
5  //  Created by Ren Chen on 2019/6/30.
6  //  Copyright © 2019 Ren Chen. All rights reserved.
7  //
8
9  #import <UIKit/UIKit.h>
10 #import "AppDelegate.h"
11
12 int main(int argc, char * argv[]) {
13     @autoreleasepool {
14         return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));
15     }
16 }
17
```

Thread 1: breakpoint...

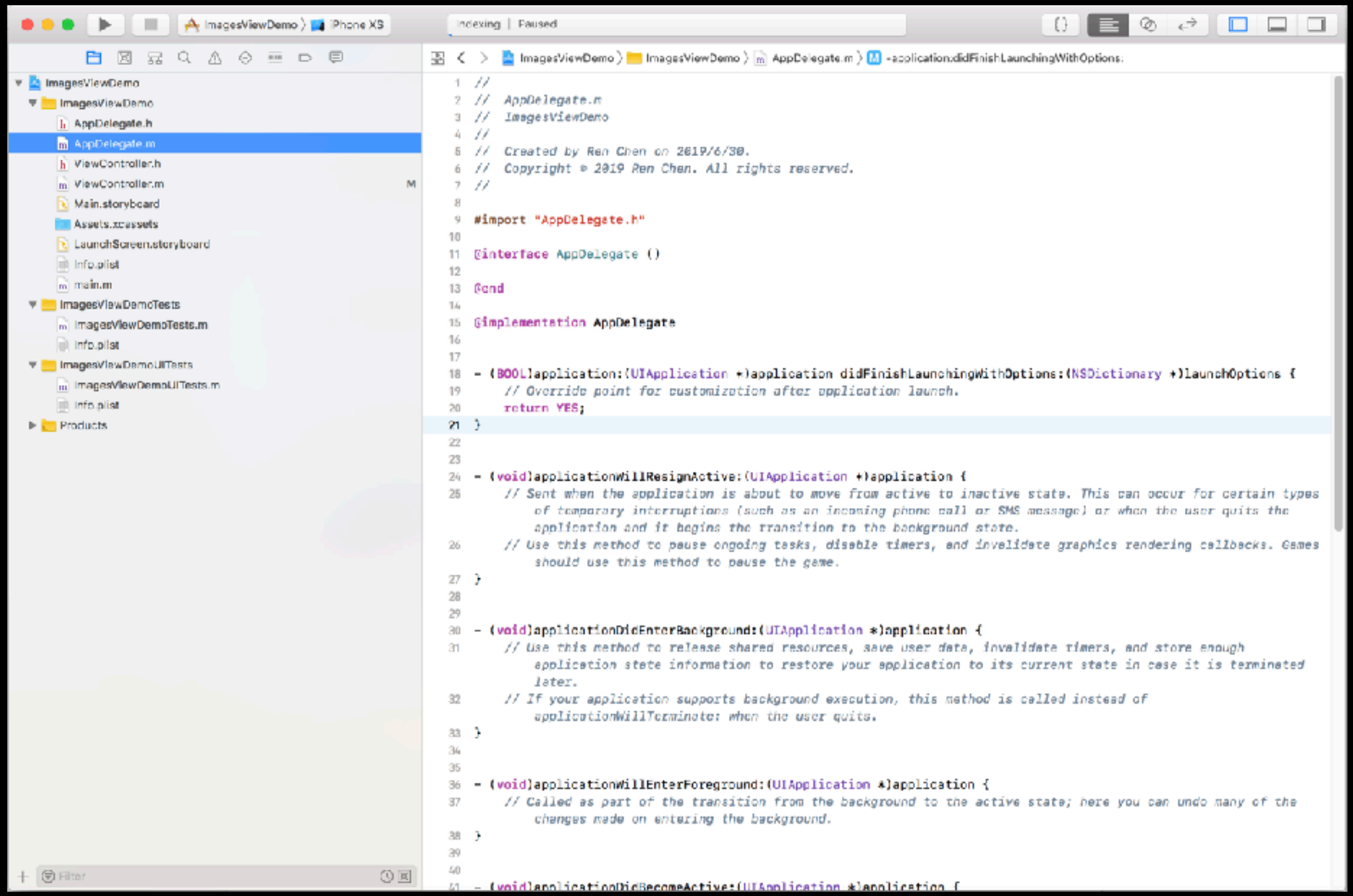
ImagesViewDemo 〉 Thread 1 〉 30 main

- argc = (int) 1
- argv = (char **) 0x7ffee5afbff8

(lldb)

创建一个简单的iOS App

ByteDance 字节跳动



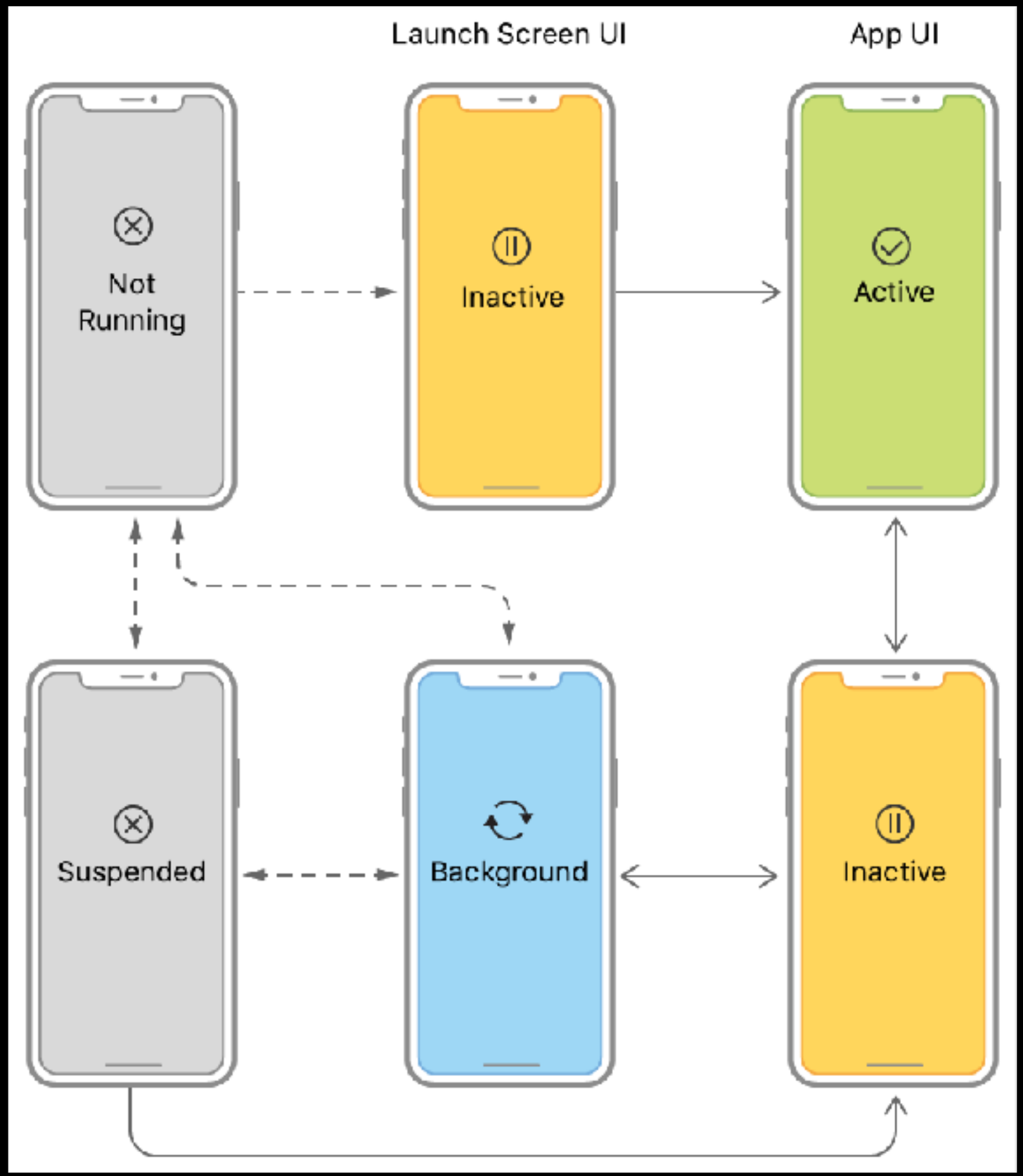
创建一个简单的iOS App

ByteDance 字节跳动

```
@implementation AppDelegate


- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    // Override point for customization after application launch.
    return YES;
}


- (void)applicationWillResignActive:(UIApplication *)application {
    // Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary
        interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the
        transition to the background state.
    // Use this method to pause ongoing tasks, disable timers, and invalidate graphics rendering callbacks. Games should use this
        method to pause the game.
}


- (void)applicationDidEnterBackground:(UIApplication *)application {
    // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information
        to restore your application to its current state in case it is terminated later.
    // If your application supports background execution, this method is called instead of applicationWillTerminate: when the user
        quits.
}


- (void)applicationWillEnterForeground:(UIApplication *)application {
    // Called as part of the transition from the background to the active state; here you can undo many of the changes made on
        entering the background.
}


- (void)applicationDidBecomeActive:(UIApplication *)application {
    // Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously
        in the background, optionally refresh the user interface.
}


- (void)applicationWillTerminate:(UIApplication *)application {
    // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:.
}


@end
```

iOS App 生命周期

iOS App 生命周期


Launch Screen UI
App UI
Not Running
Inactive
Active
Suspended
Background
Inactive

iOS App 生命周期

ByteDance 字节跳动

几种状态

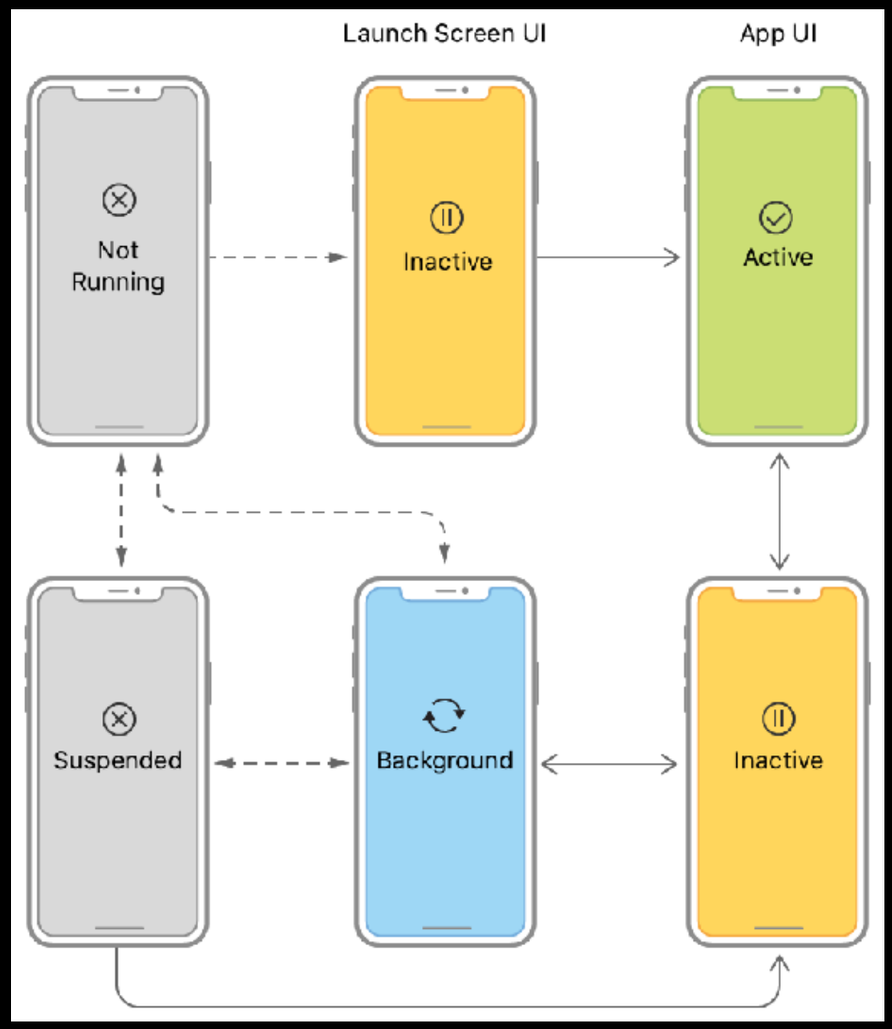
Not Running

Inactive

Active

Background

Suspended



iOS App 生命周期简介

几种状态	AppDelegate
Not Running	
Inactive	applicationWillResignActive
Active	applicationDidBecomeActive
Background	applicationWillEnterForeground
	applicationDidEnterBackground
Suspended	

iOS App 生命周期简介

双击home键	applicationWillResignActive
单击home键	applicationWillEnterForeground
进程列表选择	applicationDidBecomeActive
锁屏	applicationDidEnterBackground

UI & UIKit 简介

ByteDance 字节跳动

UIKit

Construct and manage a graphical, event-driven user interface for your iOS app

框架

```
# import <UIKit/UIKit.h>
```

window and view architecture

event handling infrastructure

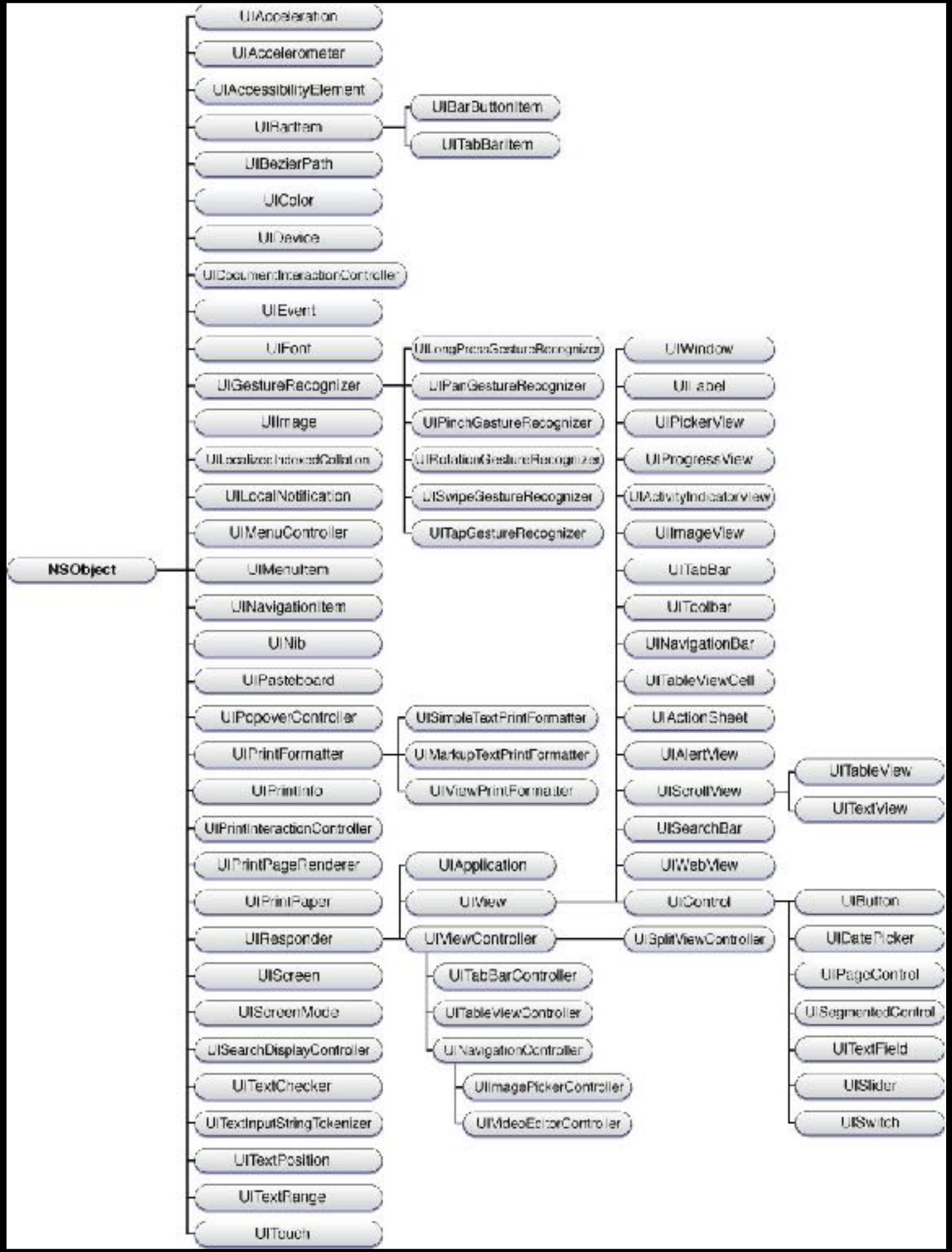
animation

document

drawing and printing

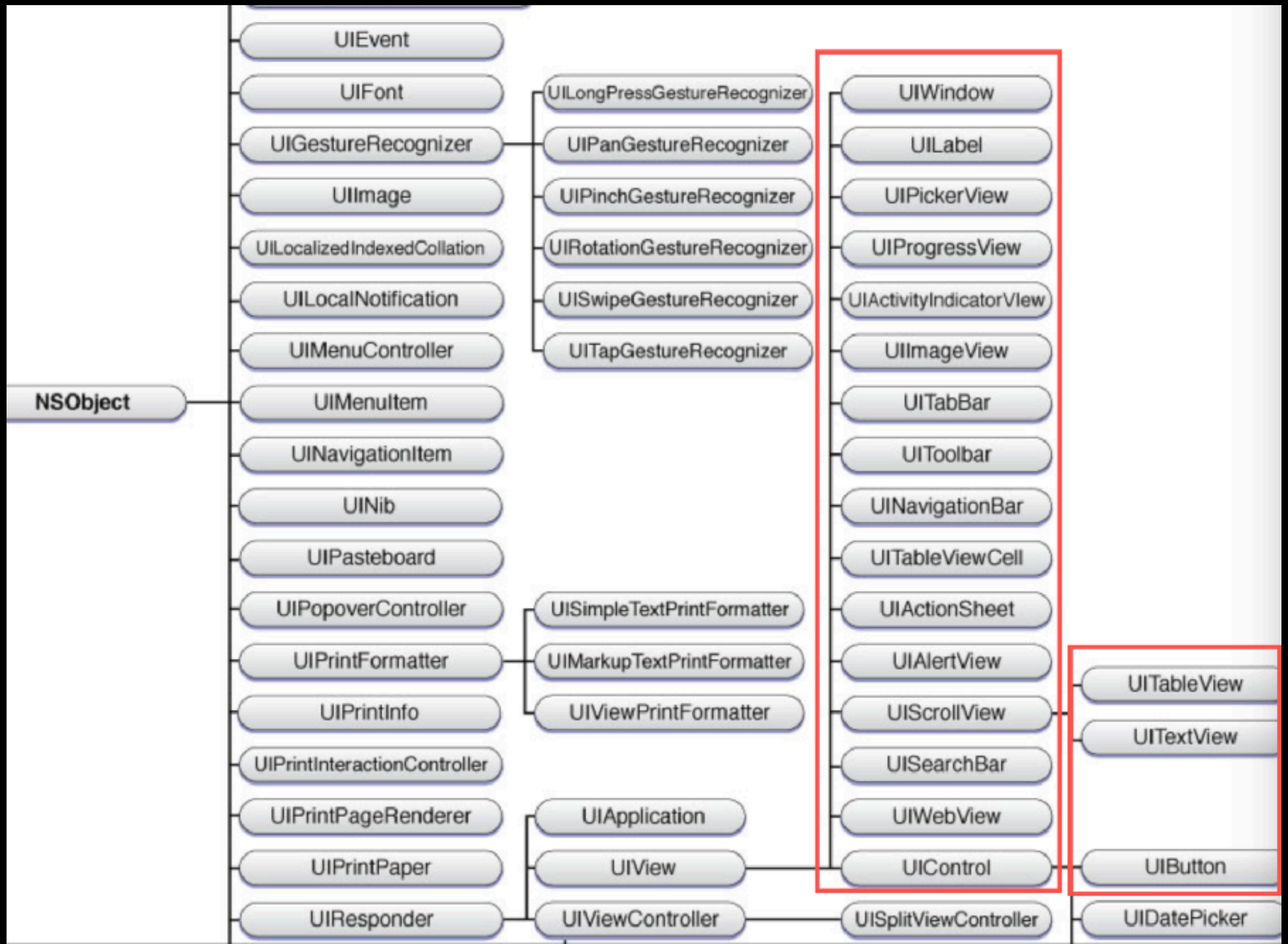
information current device

UIKit简介



ByteDance 字节跳动

UIKit简介


NSObject
UIEvent
UIFont
UIGestureRecognizer
UIImage
UILocalizedIndexedCollation
UILocalNotification
UIMenuController
UIMenuItem
UINavigationItem
UINib
UIPasteboard
UIPopoverController
UIPrintFormatter
UIPrintInfo
UIPrintInteractionController
UIPrintPageRenderer
UIPrintPaper
UIResponder
UILongPressGestureRecognizer
UIPanGestureRecognizer
UIPinchGestureRecognizer
UIRotationGestureRecognizer
UISwipeGestureRecognizer
UITapGestureRecognizer
UISimpleTextPrintFormatter
UIMarkupTextPrintFormatter
UIViewPrintFormatter
UIApplication
UIView
UIViewController
UIWindow
UILabel
UIPickerView
UIProgressView
UIActivityIndicatorView
UIImageView
UITabBar
UIToolbar
UINavigationBar
UITableViewCell
UIActionSheet
UIAlertView
UIScrollView
UISearchBar
UIWebView
UIControl
UISplitViewController
UITableView
UITextView
UIButton
UIDatePicker

ByteDance 字节跳动

UIView

An object that manages the content
for a rectangular area on the screen

UIView

```
NS_CLASS_AVAILABLE_IOS(2_0) @interface UIView : UIResponder <NSCoding,
    UIAppearance, UIAppearanceContainer, UIDynamicItem, UITraitEnvironment,
    UICoordinateSpace, UIFocusItem, UIFocusItemContainer, CALayerDelegate>

#if UIKIT_DEFINE_AS_PROPERTIES
@property(class, nonatomic, readonly) Class layerClass;
    // default is [CALayer class]. Used when creating the underlying layer
    for the view.
#else
+ (Class)layerClass;                        // default is [CALayer class].
    Used when creating the underlying layer for the view.
#endif

- (instancetype)initWithFrame:(CGRect)frame NS_DESIGNATED_INITIALIZER;
- (nullable instancetype)initWithCoder:(NSCoder *)aDecoder
    NS_DESIGNATED_INITIALIZER;
```

UIView 是 UIResponder子类

UIResponder类定义了接口应对处理事件，是所有能响应事件类的基类

UIView frame & bounds

ByteDance 字节跳动

```
@interface UIView(UIViewGeometry)

// animatable. do not use frame if view is transformed since it will not
    correctly reflect the actual location of the view. use bounds + center
    instead.
@property(nonatomic) CGRect             frame;

// use bounds/center and not frame if non-identity transform. if bounds
    dimension is odd, center may be have fractional part
@property(nonatomic) CGRect             bounds;      // default bounds is zero
    origin, frame size. animatable
@property(nonatomic) CGPoint            center;      // center is center of
    frame. animatable
@property(nonatomic) CGAffineTransform transform;    // default is
    CGAffineTransformIdentity. animatable
```

UIView frame & bounds

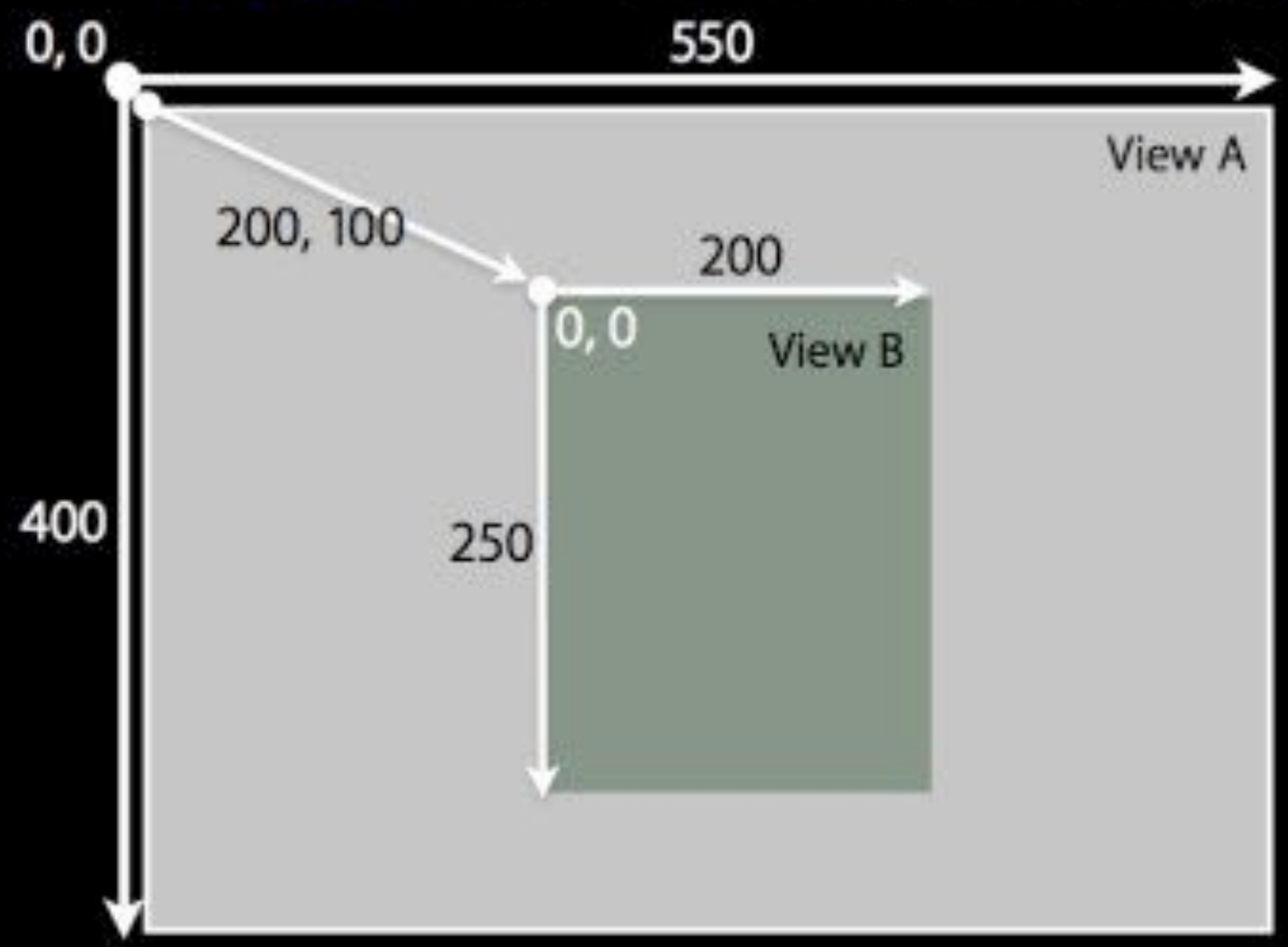
ByteDance 字节跳动

UIView 视图位置的重要概念

frame	父View坐标系统位置
bounds	本View坐标系统位置，一般起始点(0, 0)
center	父View坐标系统位置

UIView frame & bounds

- View's location and size expressed in two ways
  - Frame is in terms of superview's coordinate system
  - Bounds is in terms of local coordinate system



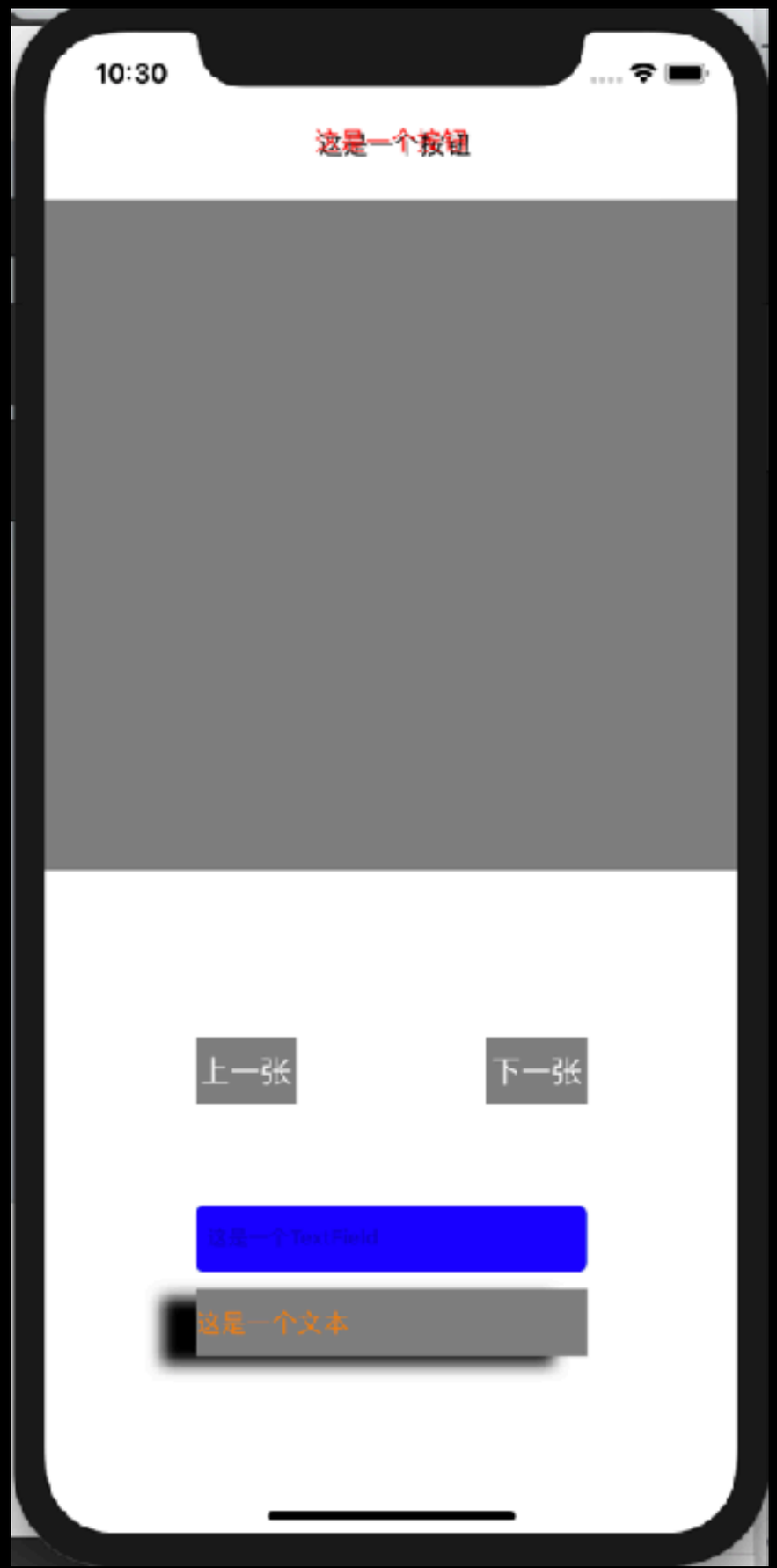
View A frame:
origin: 0, 0
size: 550 x 400

View A bounds:
origin: 0, 0
size: 550 x 400

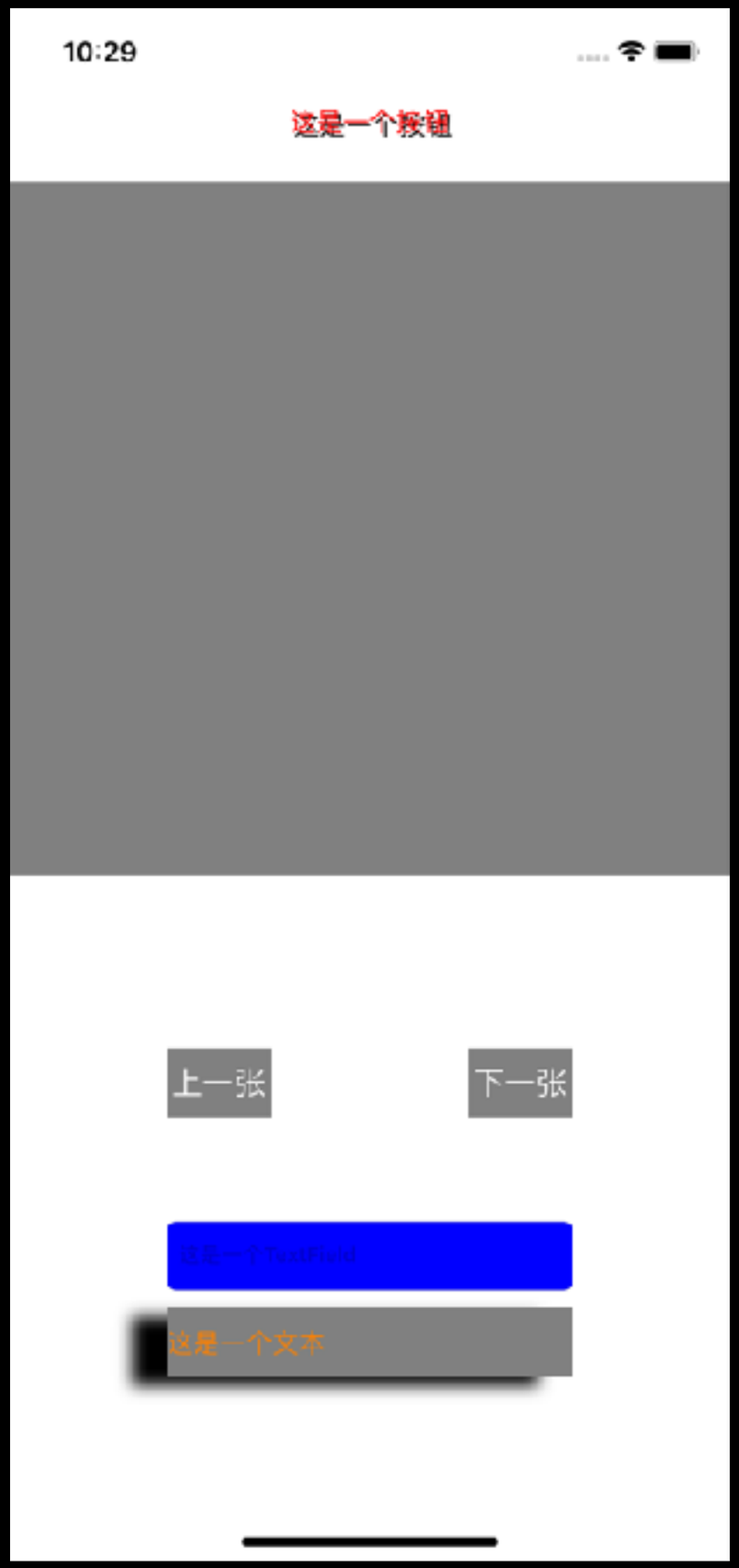
View B frame:
origin: 200, 100
size: 200 x 250

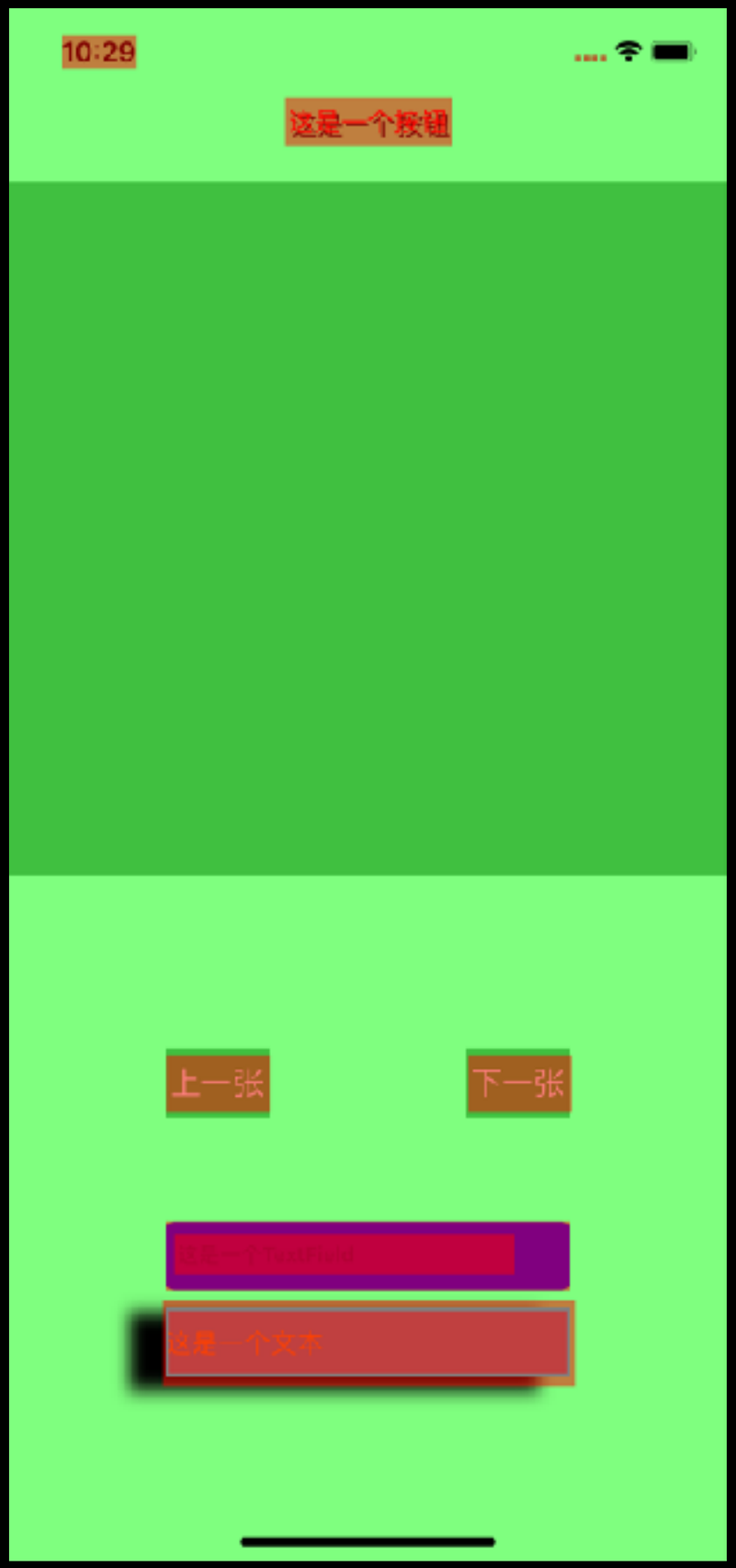
View B bounds:
origin: 0, 0
size: 200 x 250

UIView 层级结构

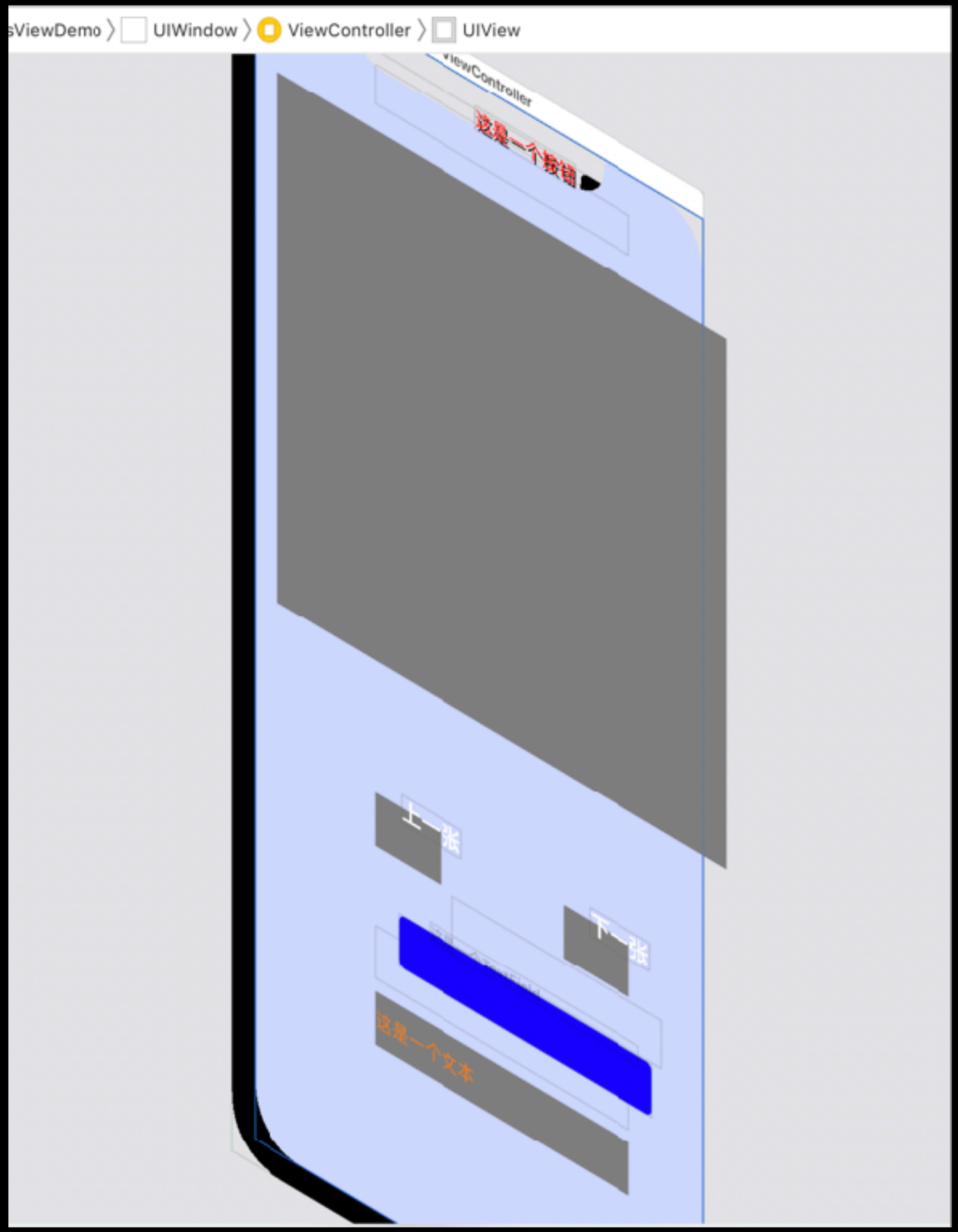


UIView 层级结构


10:29
这是一个按钮
上一张
下一张
这是一个TextField
这是一个文本

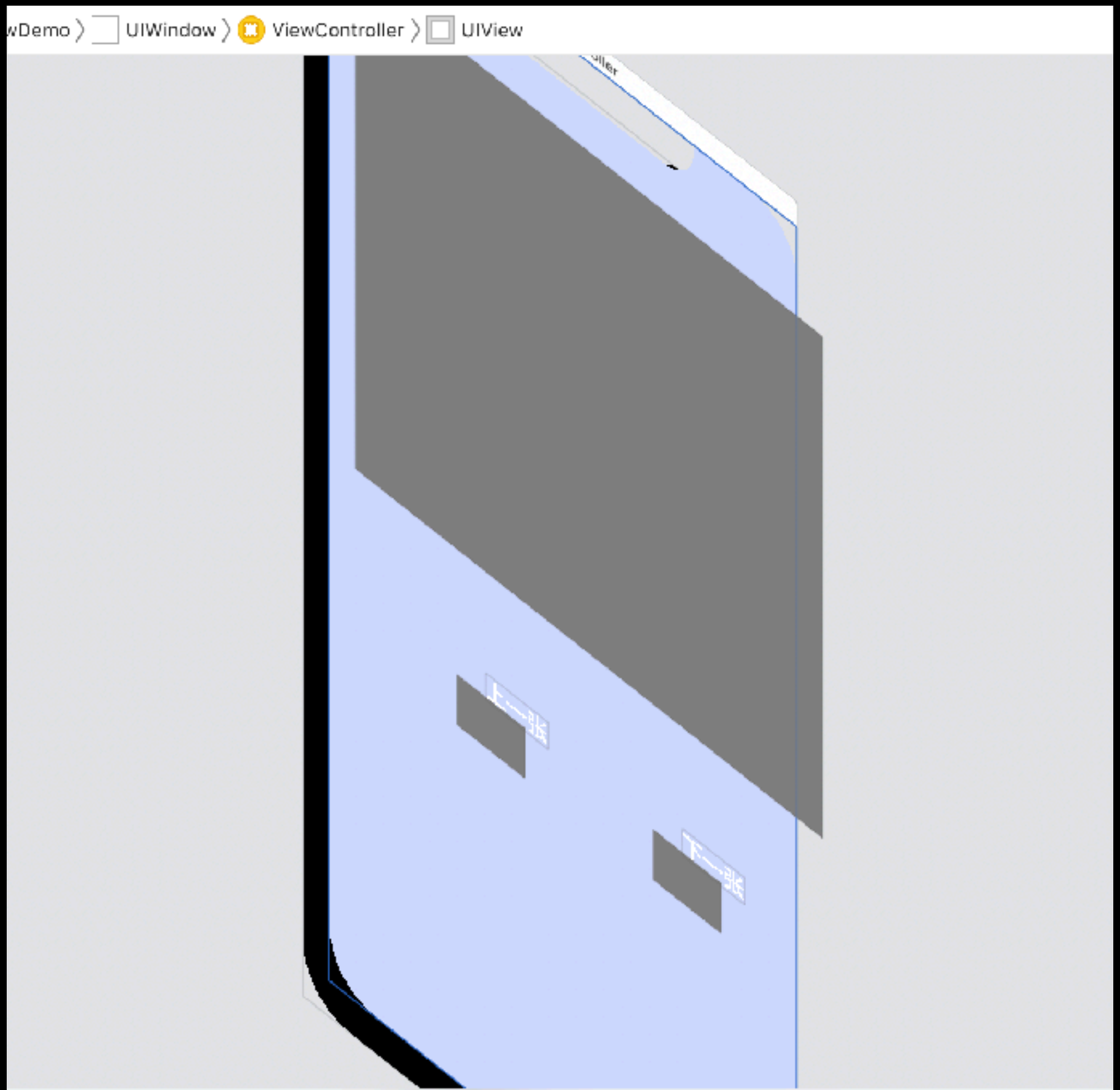

10:29
这是一个按钮
上一张
下一张
这是一个TextField
这是一个文本

UIView 层级树状结构

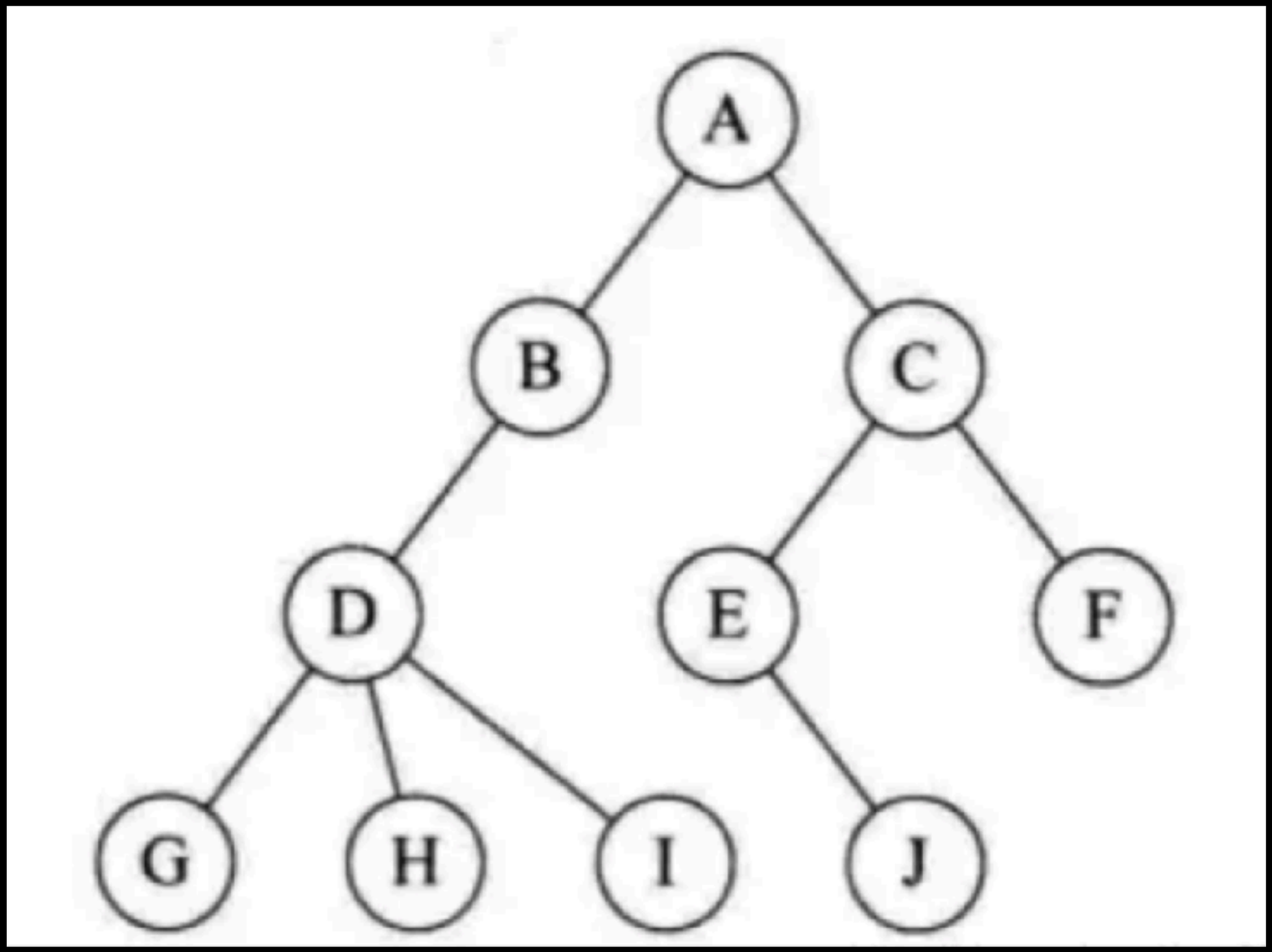


ByteDance 字节跳动

UIView 层级树状结构



UIView 层级树状结构



UIView与视图层级介绍

ByteDance 字节跳动

建立关系

addSubView

removeFromSuperView

查找

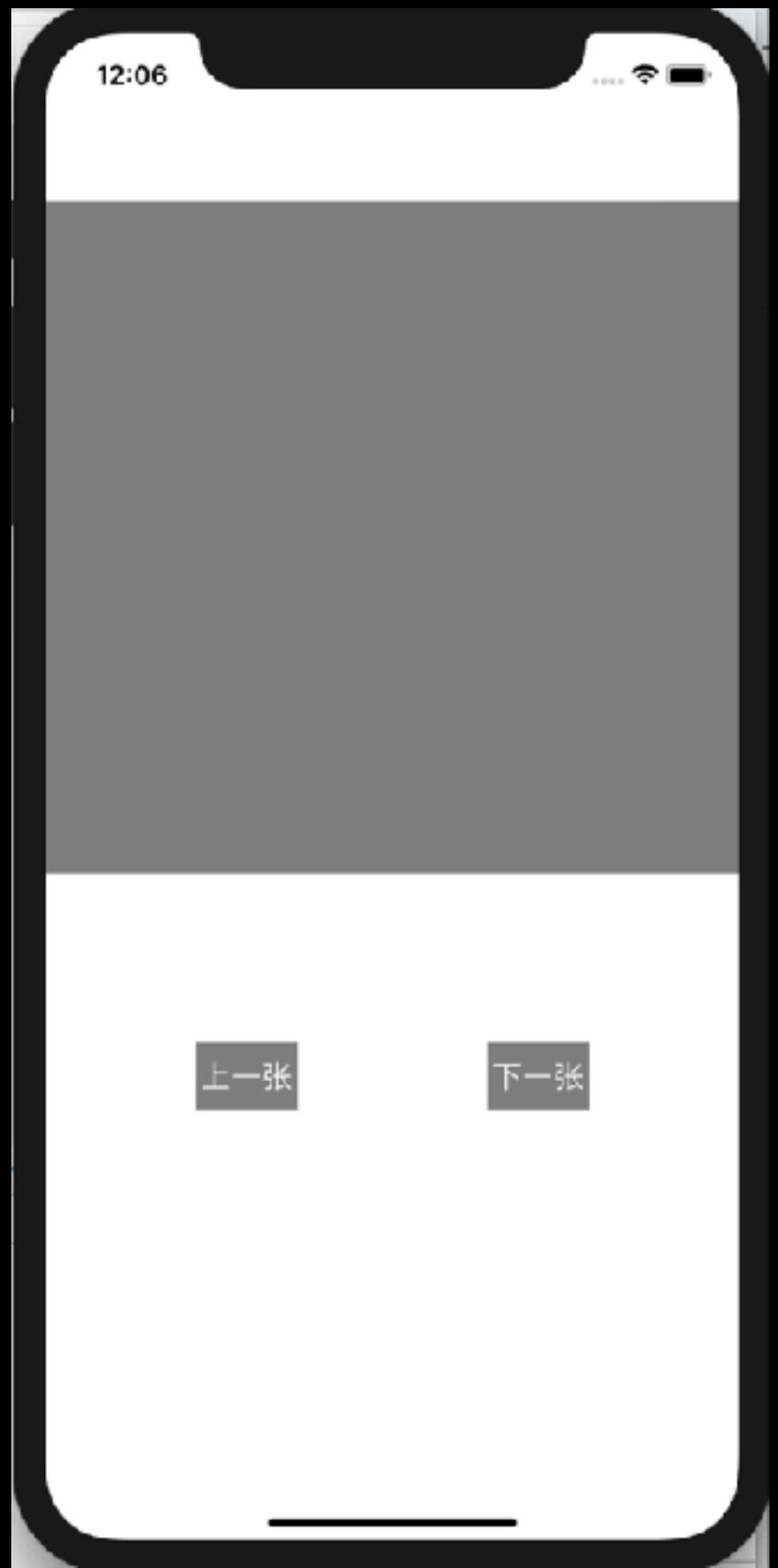
UIView tag属性，viewWithTag: 获取子视图

UIView与视图层级介绍

ByteDance 字节跳动

```
- (void)setupUI
{
    CGRect bounds = self.view.bounds;

    _imageView = [[UIImageView alloc] init];
    _nextButton = [[UIButton alloc] init];
    _previousButton = [[UIButton alloc] init];

    _imageView.backgroundColor = [UIColor grayColor];
    _nextButton.backgroundColor = [UIColor grayColor];
    _previousButton.backgroundColor = [UIColor grayColor];

    _imageView.frame = CGRectMake(0, 100, bounds.size.width, 400);
    _nextButton.frame = CGRectMake(bounds.size.width - 60 - 90, 600, 60, 40);
    _previousButton.frame = CGRectMake(90, 600, 60, 40);

    [_nextButton setTitle:@"下一张" forState:UIControlStateNormal];
    [_previousButton setTitle:@"上一张" forState:UIControlStateNormal];

    [self.view addSubview:_imageView];
    [self.view addSubview:_nextButton];
    [self.view addSubview:_previousButton];
}
```

UIView与视图层级介绍



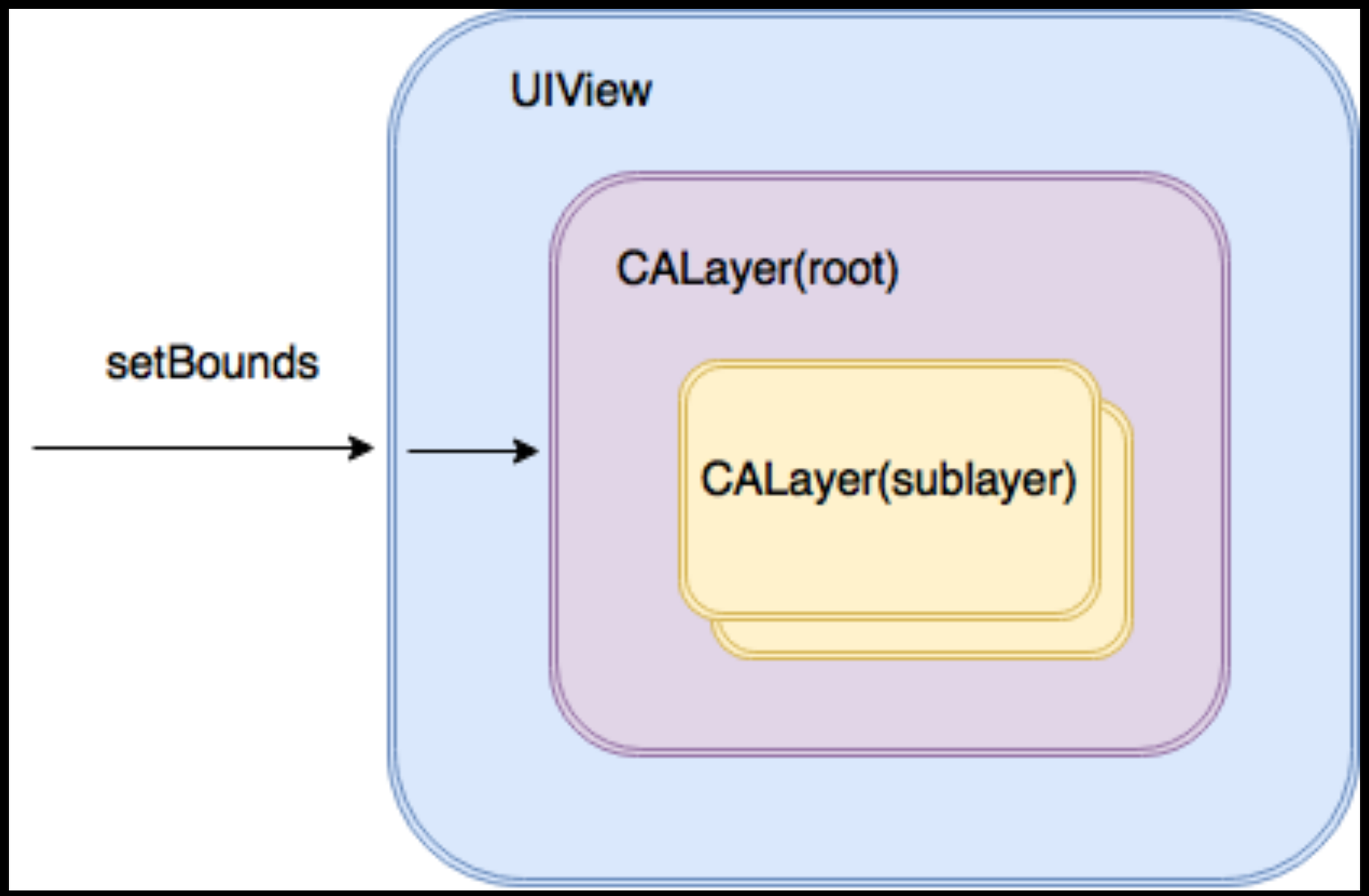
UIView layer 简介

ByteDance 字节跳动

```
#if UIKIT_DEFINE_AS_PROPERTIES
@property(class, nonatomic, readonly) Class layerClass;
    // default is [CALayer class]. Used when creating the underlying layer for
    the view.
#else
+ (Class)layerClass;                          // default is [CALayer class].
    Used when creating the underlying layer for the view.
#endif

- (instancetype)initWithFrame:(CGRect)frame NS_DESIGNATED_INITIALIZER;
- (nullable instancetype)initWithCoder:(NSCoder *)aDecoder
    NS_DESIGNATED_INITIALIZER;

@property(nonatomic,getter=isUserInteractionEnabled) BOOL
    userInteractionEnabled;  // default is YES. if set to NO, user events
    (touch, keys) are ignored and removed from the event queue.
@property(nonatomic)                                  NSInteger tag;
    // default is 0
@property(nonatomic,readonly,strong)                  CALayer  *layer;
    // returns view's layer. Will always return a non-nil value. view is
    layer's delegate
```

UIView	CALayer的高级封装
UIView	显示内容管理、事件响应
CALayer	显示内容绘制、动画过程

UIView & CALayer



ByteDance 字节跳动

布局与绘制 重要方法

layoutSubviews

layoutIfNeeded

setNeedsLayout

setNeedsDisplay

drawRect

UI & UIKit 常用控件介绍

ByteDance 字节跳动

UIButton

屏幕上的按钮控件

设置状态、文字图片、事件响应

UIKit简介 - UIButton

ByteDance 字节跳动

```
UIButton *button = [UIButton buttonWithType:UIButtonTypeSystem];
[button setFrame:CGRectMake(125, 50, 70, 30)];
[button setTitle:@"这是一个按钮" forState:UIControlStateNormal];
// 按钮默认状态设置文本颜色为红色
[button setTitleColor:[UIColor redColor] forState:UIControlStateNormal];
// 按钮默认状态设置文本阴影颜色为黑色
[button setTitleShadowColor:[UIColor blackColor] forState:UIControlStateNormal];
// 按钮文本的阴影设置偏移量
[button.titleLabel setShadowOffset: CGSizeMake(2, 2)];
[button addTarget:self action:@selector(showNextImage)
    forControlEvents:UIControlEventTouchUpInside];

[self.view addSubview:button];
```

UILabel

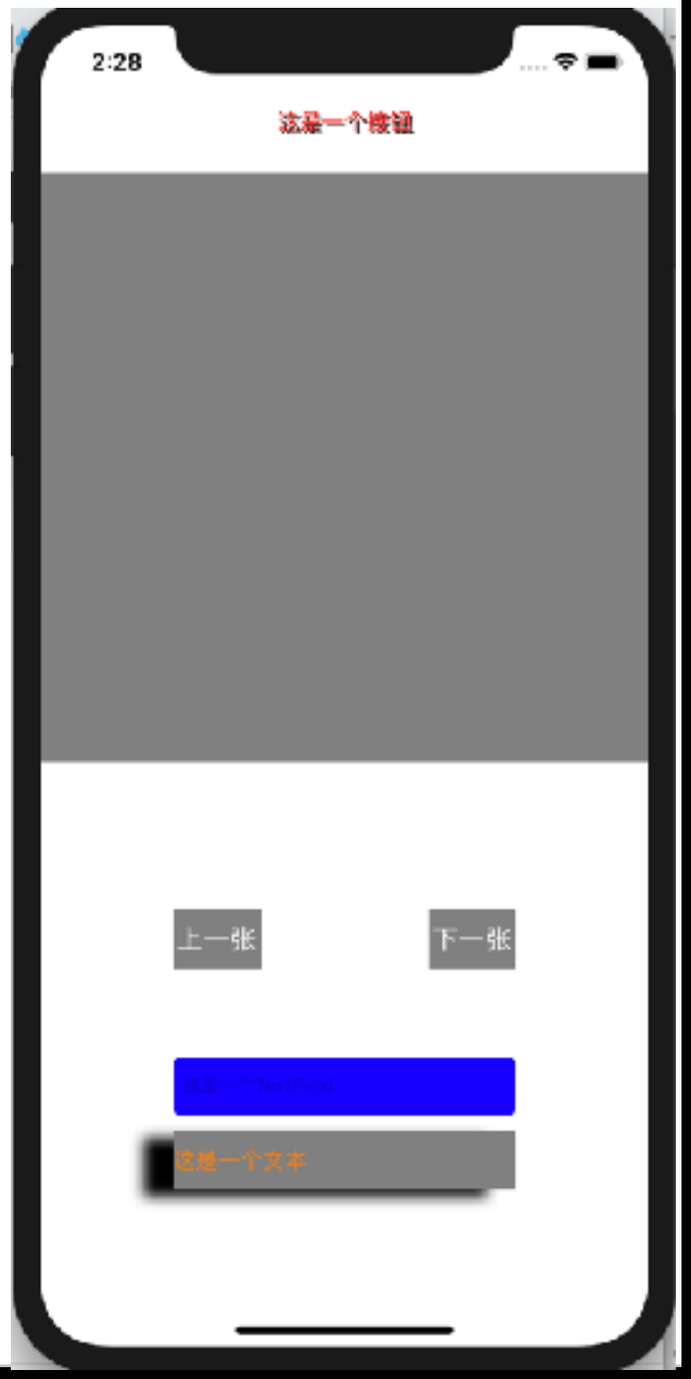
显示控件，标签

单纯用以显示文字的基本控件

UIKit简介 - UILabel

ByteDance 字节跳动

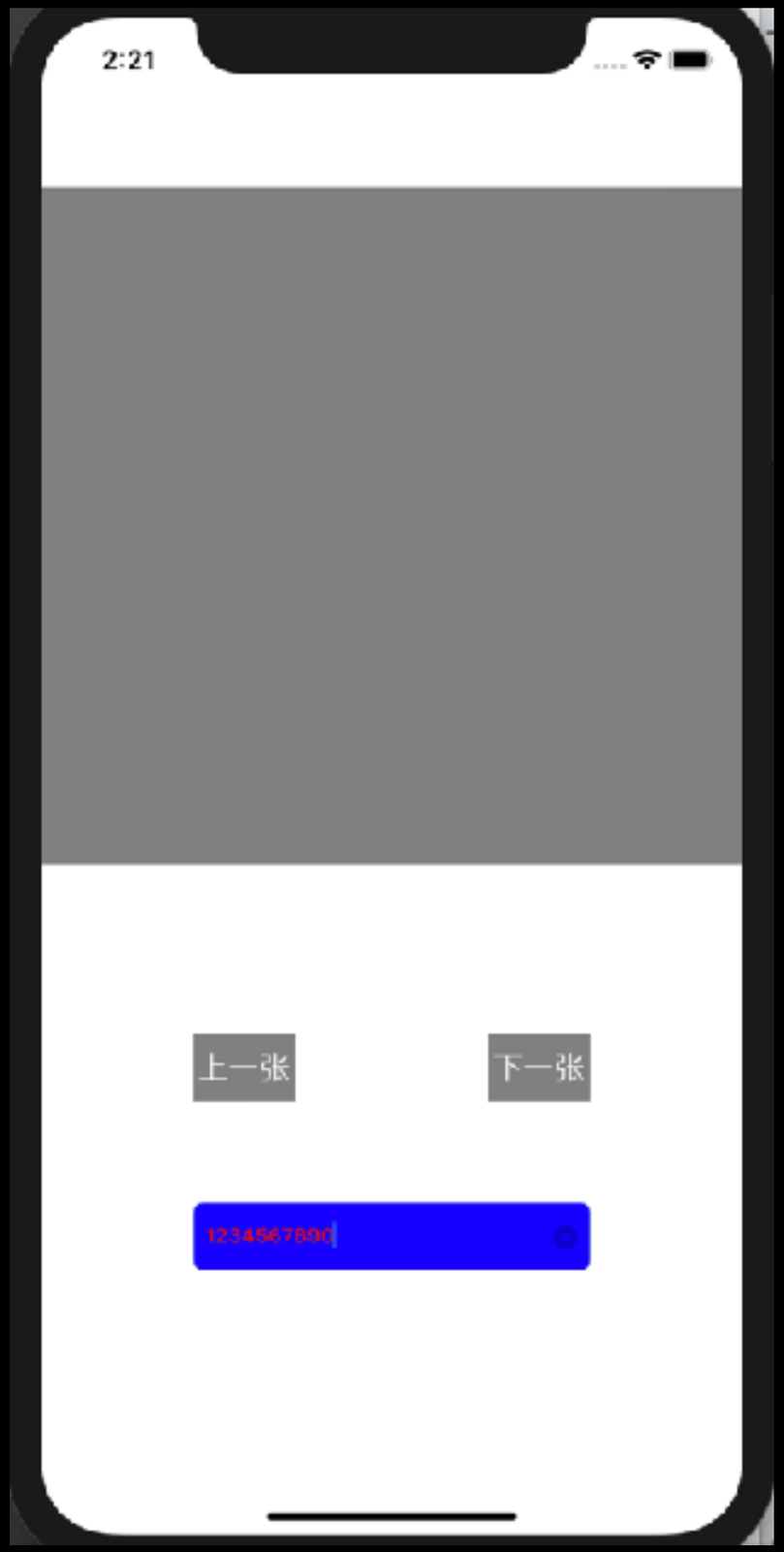
```
UILabel *label = [[UILabel alloc] initWithFrame:CGRectMake(90, 750, bounds.size.width - 180,
    40)];
label.backgroundColor = [UIColor grayColor];
// 设置显示的内容
label.text = @"这是一个文本";
// 设置字体颜色
label.textColor = [UIColor orangeColor];
// 设置字体和字号
label.font = [UIFont systemFontOfSize:15];
// 设置多行显示
label.numberOfLines = 0;
// 设置换行的方式
label.lineBreakMode = NSLineBreakByCharWrapping;
// 设置对齐方式
label.textAlignment = NSTextAlignmentLeft;

// 设置阴影
label.layer.shadowColor = [UIColor blackColor].CGColor;
label.layer.shadowOffset = CGSizeMake(-20, 5);
label.layer.shadowRadius = 5;
label.layer.shadowOpacity = 1;
// 添加边框，设置颜色与宽度
label.layer.borderColor = [[UIColor grayColor] CGColor];
label.layer.borderWidth = 2;

[self.view addSubview:label];
```



UITextField

可编辑本文控件

无法换行，只能一行显示



```
UITextField *textField = [[UITextField alloc] initWithFrame:CGRectMake(90, 700,
    bounds.size.width - 180, 40)];
textField.tag = 100;
// 更改背景颜色
textField.backgroundColor = [UIColor blueColor];
// 边框类型
textField.borderStyle = UITextBorderStyleRoundedRect;
// 字体
textField.font = [UIFont boldSystemFontOfSize:12.0];
// 字体颜色
textField.textColor = [UIColor grayColor];
// 对齐方式
textField.textAlignment = NSTextAlignmentLeft;
// 垂直对齐
textField.contentVerticalAlignment = UIControlContentVerticalAlignmentCenter;
// 文字缩放
textField.adjustsFontSizeToFitWidth = YES;
// 缩放后最小字号
textField.minimumFontSize = 40.0;
// 文本
//textField.text = @"输入文本";
// 占位文字
textField.placeholder = @"这是一个TextField";
//清空按钮
textField.clearButtonMode = UITextFieldViewModeAlways;
// 当编辑时清空
//textField.clearsOnBeginEditing = YES;
// 自动大写
//textField.autocapitalizationType = UITextAutocapitalizationTypeAllCharacters;
// 单词检测 是否是单词 显示下划线
textField.autocorrectionType = UITextAutocorrectionTypeNo;
//textField.delegate = self;
textField.borderStyle = UITextBorderStyleRoundedRect;

[self.view addSubview:textField];
```

ByteDance 字节跳动

UIImage & UIImageView

UIImage	图像类
UIImageView	图像视图控件

ByteDance 字节跳动

UIImage

```
[UIImage imageNamed:@“sample.png”];

NSString *path = [[NSBundle mainBundle]
pathForResource:@“sample” ofType:@“png”];
myImage = [UIImage imageWithContentsOfFile:path];
```

ByteDance 字节跳动

UIImageView

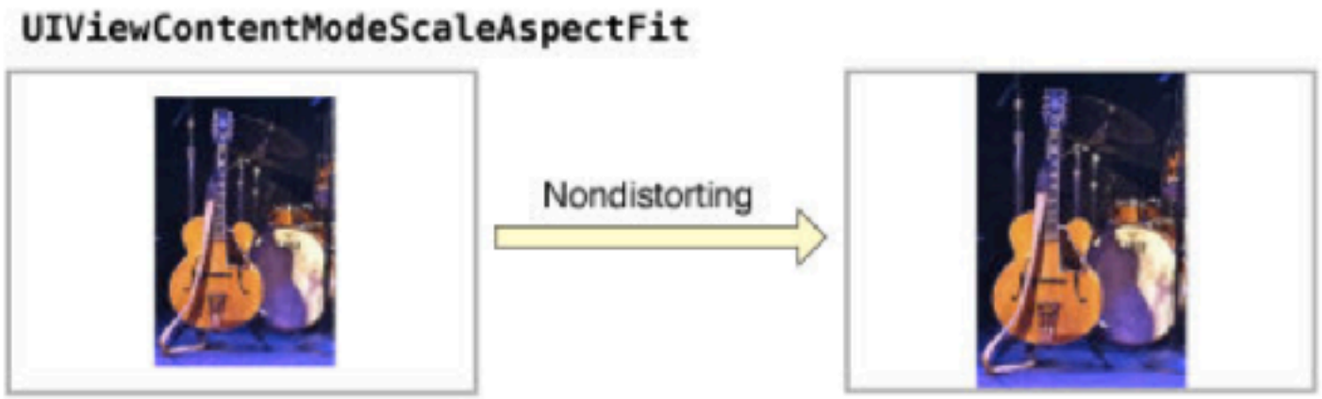
```
[[UIImageView alloc] initWithImage:image];
```

UIImageView contentMode

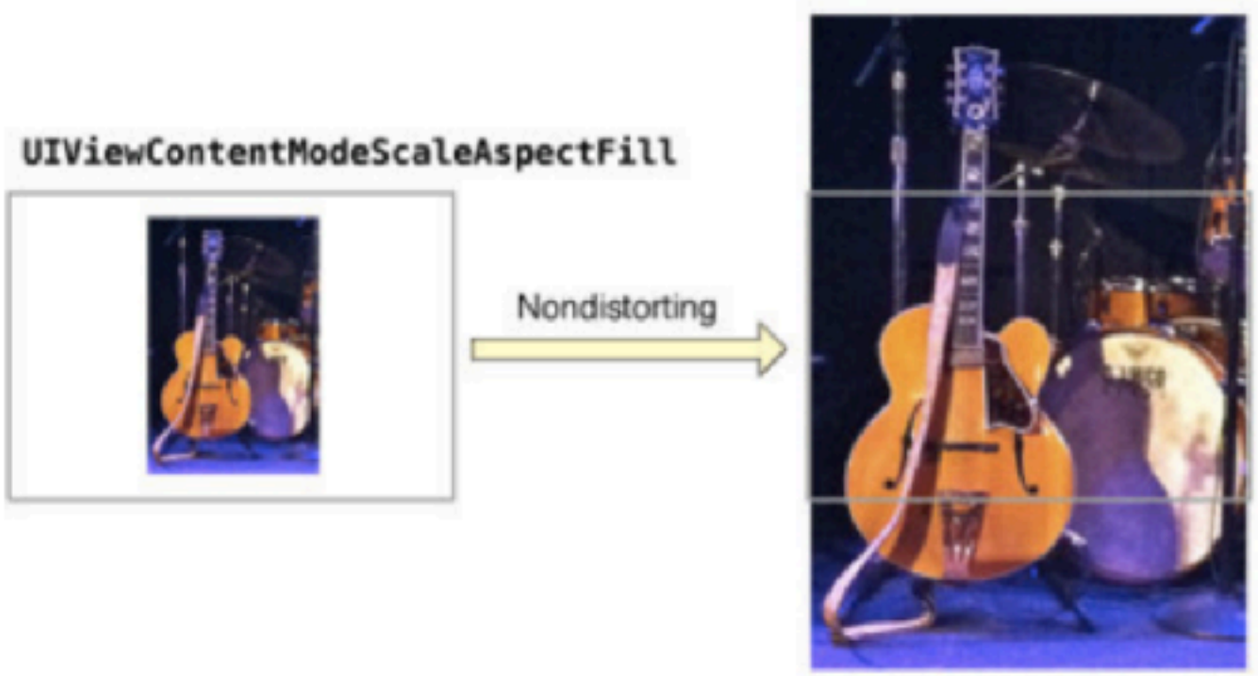
```
typedef NS_ENUM(NSInteger, UIViewContentMode) {
    UIViewContentModeScaleToFill,
    UIViewContentModeScaleAspectFit,      // contents scaled to fit
        with fixed aspect. remainder is transparent
    UIViewContentModeScaleAspectFill,     // contents scaled to fill
        with fixed aspect. some portion of content may be clipped.
    UIViewContentModeRedraw,              // redraw on bounds change
        (calls -setNeedsDisplay)
    UIViewContentModeCenter,              // contents remain same
        size. positioned adjusted.
    UIViewContentModeTop,
    UIViewContentModeBottom,
    UIViewContentModeLeft,
    UIViewContentModeRight,
    UIViewContentModeTopLeft,
    UIViewContentModeTopRight,
    UIViewContentModeBottomLeft,
    UIViewContentModeBottomRight,
};
```

UIViewContentModeScaleToFill模式会导致图片变形。例如:

UIViewContentModeScaleAspectFit会保证图片比例不变，而且全部显示在ImageView中，



UIViewContentModeScaleAspectFill也会证图片比例不变，但是是填充整个ImageView的



事件、手势与响应原理

UIEvent 代表在iOS系统的事件

主要三种类型事件

触摸事件	Touch Events
运动事件	Motion Events
远程控制的事件	Remote Events

ByteDance 字节跳动

手势识别器UIGestureRecognizer

iOS系统手势识别抽象基类

#UITapGestureRecognizer

单次或多次屏幕点击手势

#UISwipeGestureRecognizer

屏幕上下左右滑动手势

UI事件、手势识别

ByteDance 字节跳动

```
    _imageView.userInteractionEnabled = YES;
    for (NSInteger i = 0; i < 4; i++) {
        UISwipeGestureRecognizer *recognizer = [[UISwipeGestureRecognizer alloc] initWithTarget:self
            action:@selector(handleSwipe:)];
        recognizer.numberOfTouchesRequired = 1;
        // 该手势处理器只处理 1 << i 方向的轻扫手势
        recognizer.direction = 1 << i;
        [_imageView addGestureRecognizer:recognizer];
    }
}

-(void)handleSwipe:(UISwipeGestureRecognizer *)recognizer {
    if (recognizer.direction==UISwipeGestureRecognizerDirectionUp) {
        NSLog(@"swipe up");
        //[self showNextImage];
    }

    if (recognizer.direction==UISwipeGestureRecognizerDirectionDown) {
        NSLog(@"swipe down");
        //[self showPreviousImage];
    }

    if (recognizer.direction==UISwipeGestureRecognizerDirectionLeft) {
        NSLog(@"swipe left");
        [self showNextImage];
    }

    if (recognizer.direction==UISwipeGestureRecognizerDirectionRight) {
        NSLog(@"swipe right");
        [self showPreviousImage];
    }
}
```

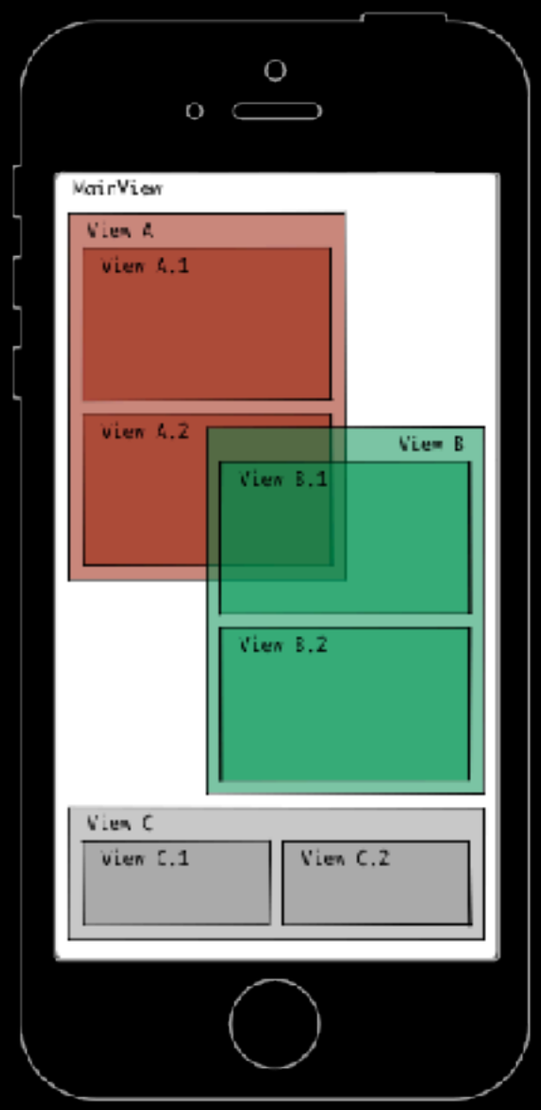
Touch Events (触摸事件响应)

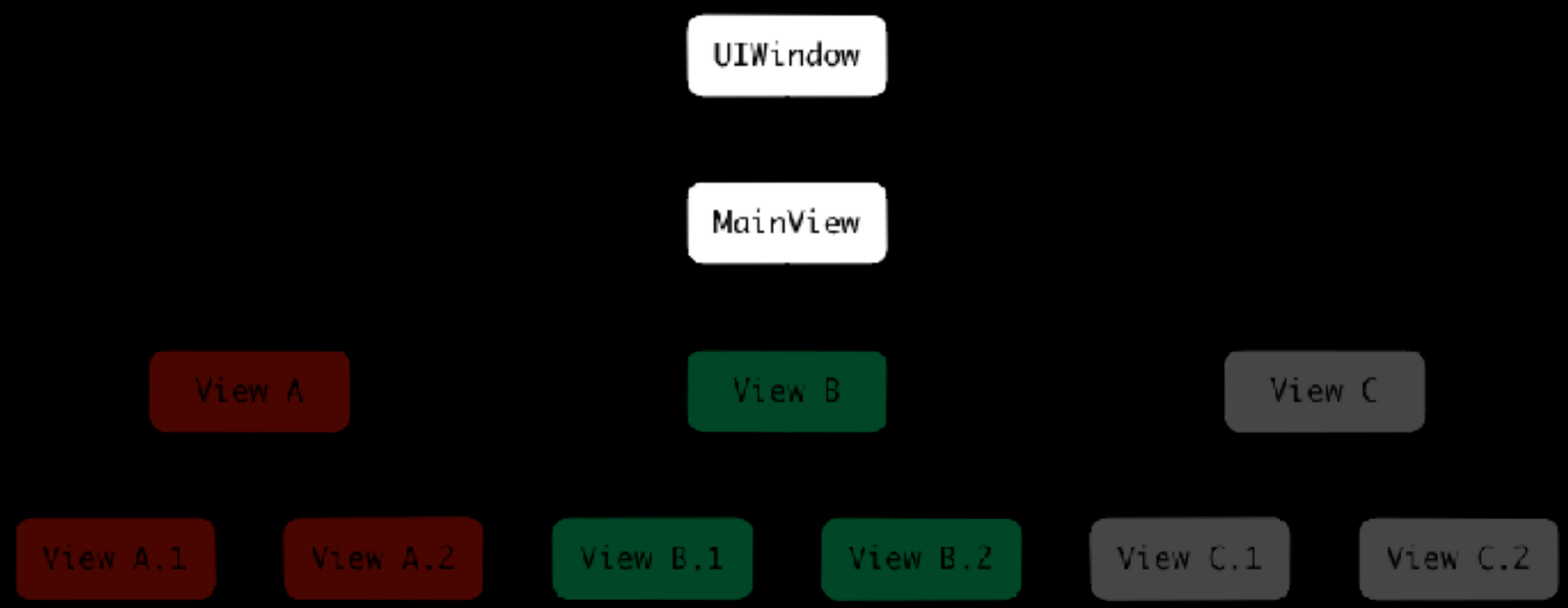
Touch Events 事件过程分为传递和响应两个阶段

传递：　触摸屏幕找到最合适的View

响应：　找到最适合的View后，继续找能响应事件的View

Touch Events (触摸事件响应)





Touch Events (触摸事件响应)

ByteDance 字节跳动

事件的分发和传递

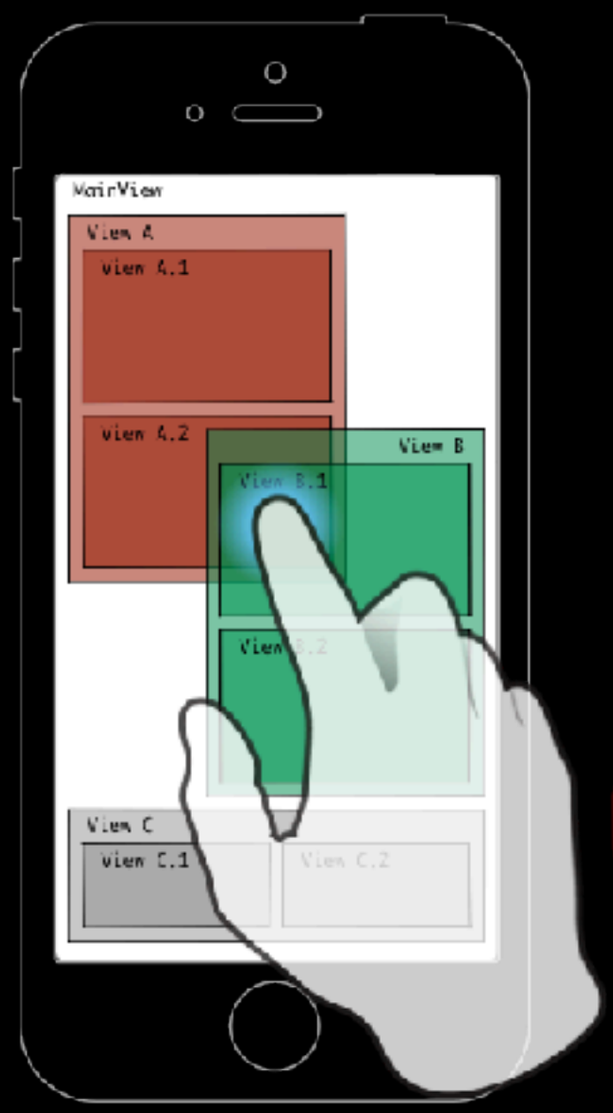
1. 当iOS程序中发生触摸事件后，系统会将事件加入到UIApplication管理的一个任务队列
2. UIApplication将处于任务队列最前端的事件向下分发，先是UIWindow
3. UIWindow将事件向下分发，即UIView
4. UIView首先看自己是否能处理事件，触摸点是否在范围内；如果能，那么继续寻找子视图看能否处理
5. 遍历子控件，重复以上两步
6. 如果没有找到，那么自己就是事件处理者
7. 如果自己不能处理，那么不做任何处理

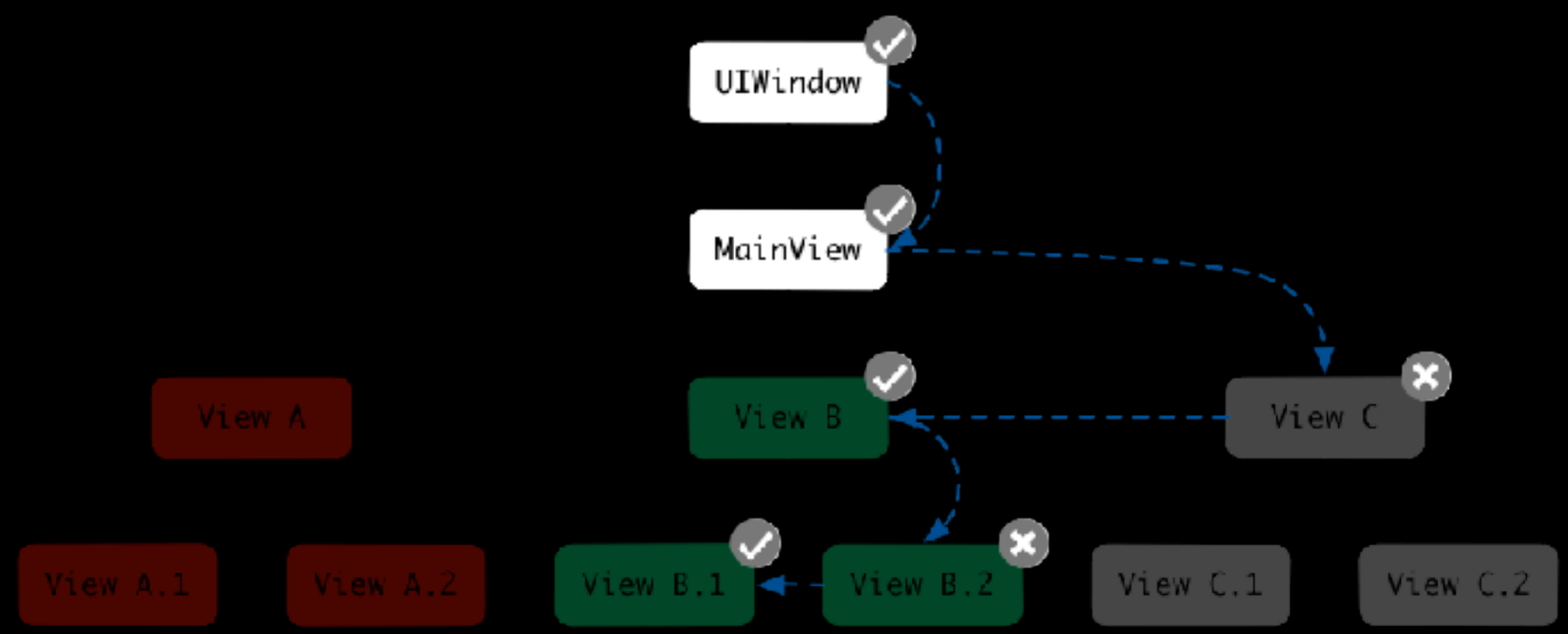
ByteDance 字节跳动

UIView 不接受事件响应的场景

1. alpha < 0.01
2. userInteractionEnabled = NO
3. hidden = YES

Touch Events (触摸事件响应)





Touch Events (触摸事件响应)

递归是向界面的根节点UIWindow发送hitTest:withEvent:消息开始

从这个消息返回的是一个UIView，也就是手指当前位置最前面的那个 hittest view

Touch Events (触摸事件响应)

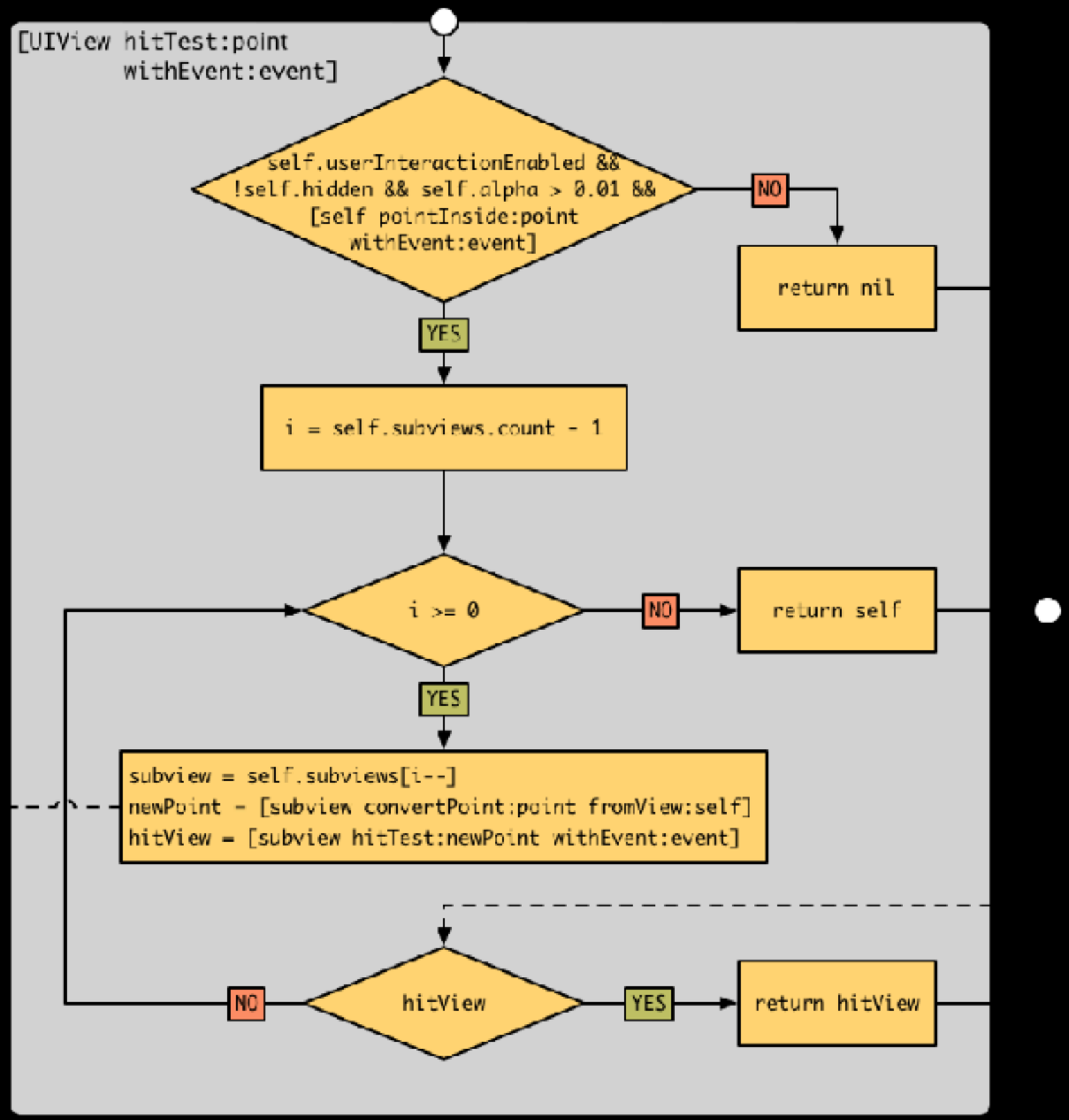
当向UIWindow发送hitTest:withEvent:消息时，hitTest:withEvent:里面所做的事，就是判断当前的点击位置是否在window里面

如果在则遍历window的subview然后依次对subview发送hitTest:withEvent:消息(注意这里给subview发送消息是根据当前subview的index顺序，index越大就越先被访问)

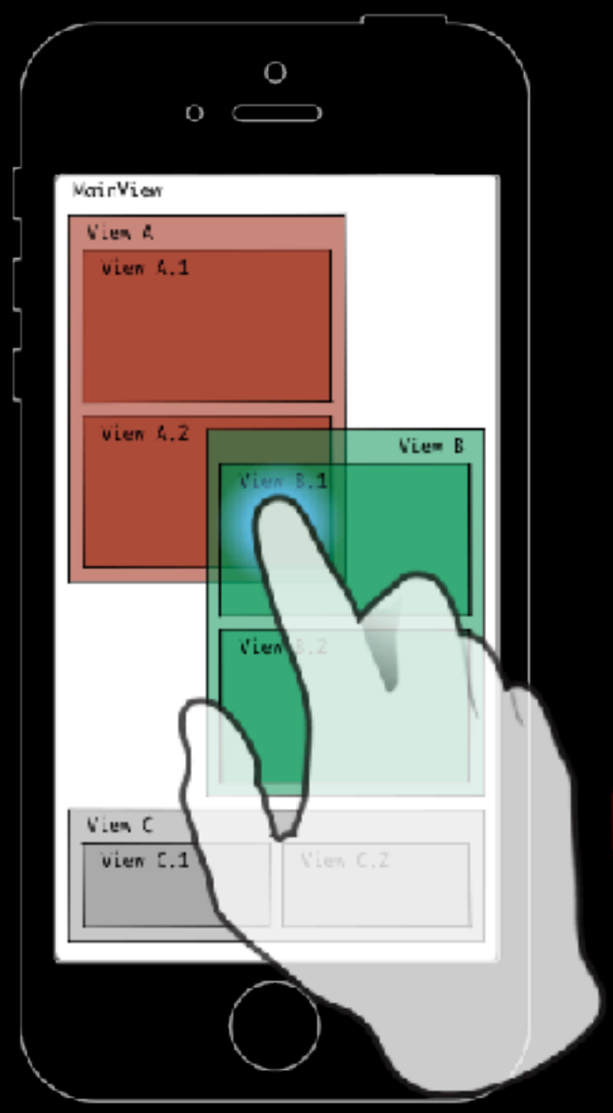
如果当前的point没有在view上面，那么这个view的subview也就不会被遍历

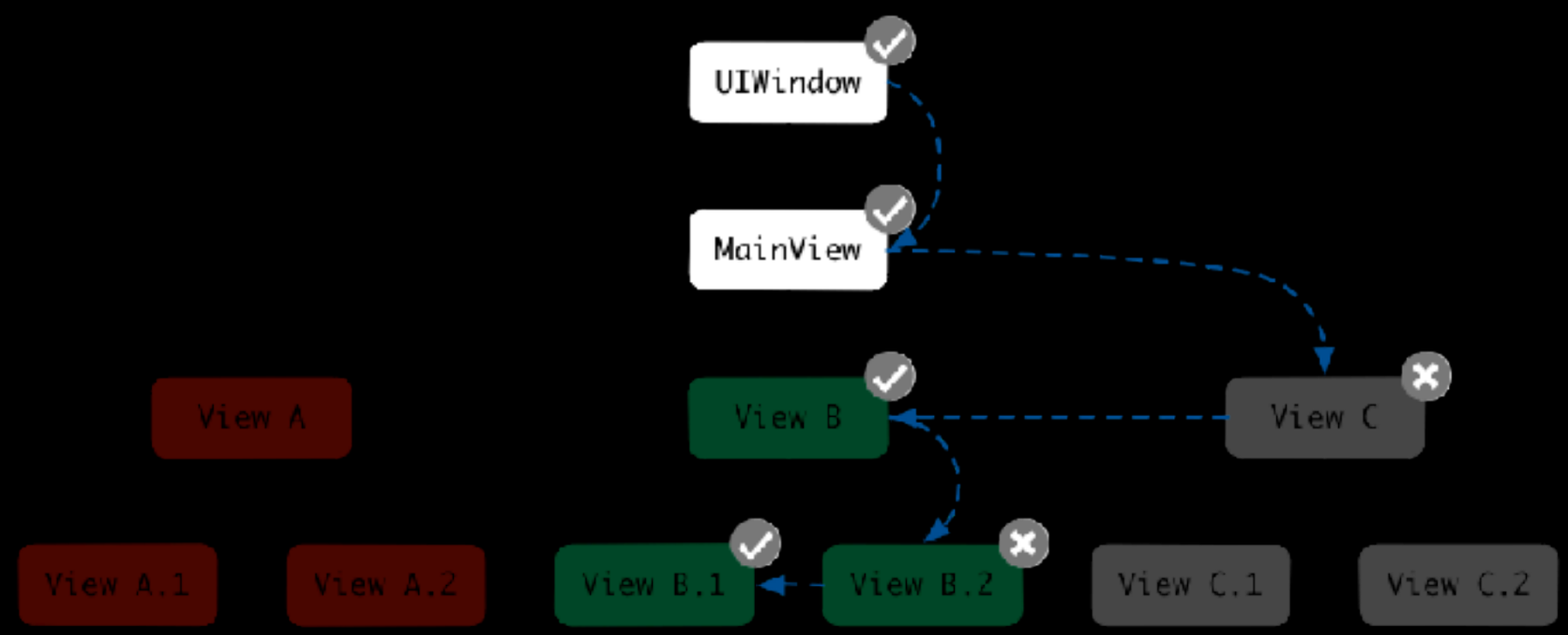
Touch Events (触摸事件响应)

[UIWindow hitTest:point withEvent:event]



Touch Events (触摸事件响应)





Demo练习

Summary

ByteDance
字节跳动