IOS学习笔记

Apple Framework

UIAppDelegate

当Apple程序运行时会创建App实例，UIAppDelegate的本质是一个抽象接口类，创建App之后将AppDelegate绑定到App实例当中，App程序去调用AppDelegate中对应接口的实现。

OC入口函数

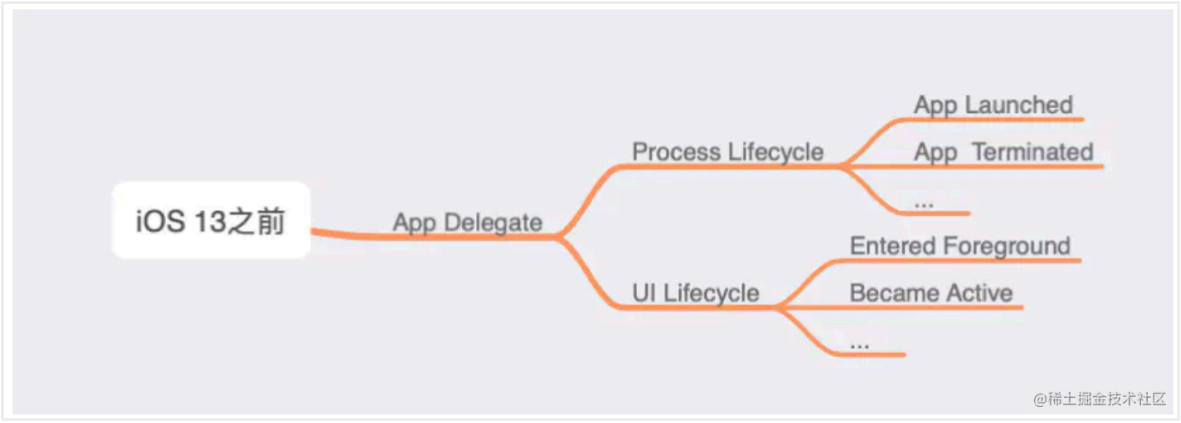
```
int main(int argc, char * argv[]) {
    NSString * appDelegateClassName;
    @autoreleasepool {
        appDelegateClassName = NSStringFromClass([MyAppDelegate class]);
    }
    return UIApplicationMain(argc, argv, nil, appDelegateClassName);
}
```

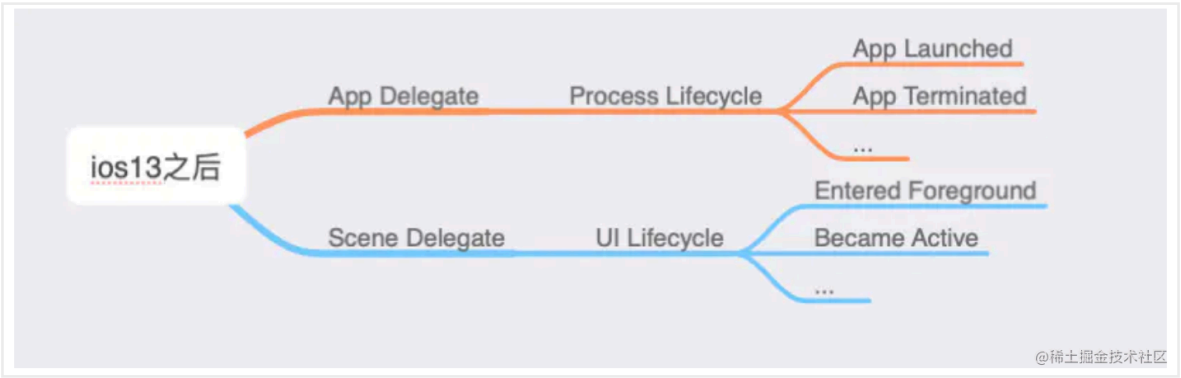
C++入口函数

```
int main( int argc, char* argv[] )
{
    NS::AutoreleasePool* pAutoreleasePool = NS::AutoreleasePool::alloc()-
>init();
    MyAppDelegate del;
    NS::Application* pSharedApplication = NS::Application::sharedApplication();
    pSharedApplication->setDelegate( &del );
    pSharedApplication->run();
    pAutoreleasePool->release();
    return 0;
}
```

UISceneDelegate

ios13之后，Apple官方新增UISceneDelegate抽象接口类用于实现UI生命周期，而UIAppDelegate只负责APP的生命周期管理，在此之前，APP的生命周期和UI生命周期是由UIAppDelegate全权负责的。



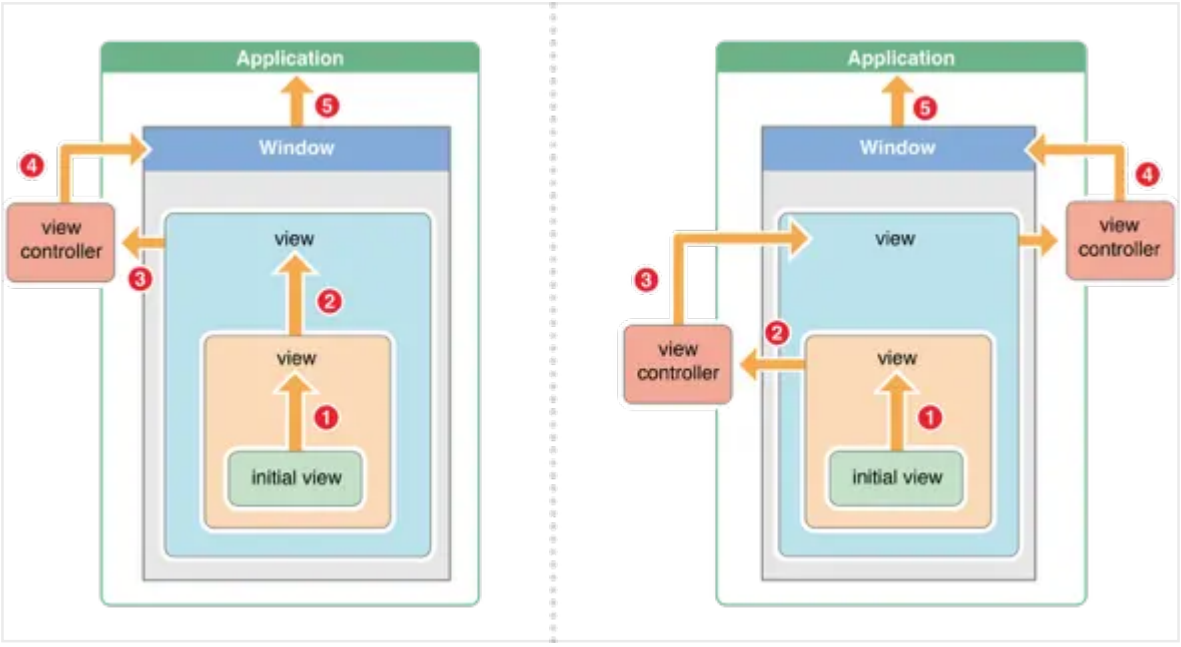


在ios13之前，负责初始化应用程序中Window在UIAppDelegate中的(BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions函数中进行。
而在ios13之后，负责初始化应用程序中Window在UISceneDelegate中的(void)scene:(UIScene *)scene willConnectToSession:(UISceneSession *)session options:(UISceneConnectionOptions *)connectionOptions函数中进行。

Apple Event

事件传递和响应

IOS事件由响应连和传递链构成。一般事件先通过传递链传递下去。如果最上层不能响应，那么一层一层通过响应链找到能响应的UIResponse。iOS 中只有继承了UIResponse的对象才能够接受处理事件。UIResponse是响应对象的基类，定义了处理上述各种事件的接口。常见的子类有：UIView，UIViewController，UIApplication和UIApplicationDelegate。
iOS中的view之间逐层叠加，当点击了屏幕上的某个view时，这个点击动作会由硬件层传导到操作系统并生成一个事件(event)，这个事件将会进入Application中的事件队列当中，需要处理的事件会通过传递链从系统向最上层view传递，Application -> window -> root view -> ... -> first view，找到响应的view之后，通过响应连将被响应view向系统传递，first view -> super view -> ... -> view controller -> window -> Application -> AppDelegate



UIResponse

UIResponder提供了许多方法来处理事件，其中事件可以分为三大类型：

- 触摸事件：

```
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event;
```

- 加速计事件：

```
- (void)motionBegan:(UIEventSubtype)motion withEvent:(UIEvent *)event;
- (void)motionEnded:(UIEventSubtype)motion withEvent:(UIEvent *)event;
- (void)motionCancelled:(UIEventSubtype)motion withEvent:(UIEvent *)event;
```

- 远程控制事件

```
- (void)remoteControlReceivedWithEvent:(UIEvent *)event;
```

UIEvent

UIEvent记录事件产生的时刻和类型，每产生一个事件就会产生一个UIEvent对象。

UITouch

用户用手指触摸屏幕时，会创建一个与手指相关联的UITouch对象，一根手指对应一个UITouch对象。其中保存着跟手指相关的信息，比如触摸的位置、时间、阶段等。当手指移动时，系统会更新同一个UITouch对象，使之能够一直保存该手指在的触摸位置。当手指离开屏幕时，系统会销毁相应的UITouch对象。

UIKit(Apple UI 框架)

UIView

UIView是基本类，提供UI的渲染，其中包含了addSubView等UI基本功能，用于与用户交互和展示界面功能，对应MVC框架的V。

UIView响应事件时，会从上层view向下层view传递被响应的view，可以通过重写-(nullable UIView *)hitTest:(CGPoint)point withEvent:(nullable UIEvent *)event方法来修改被响应的view。

如果UIView被隐藏(hidden == true)，禁用用户操作(userInteractionEnabled == false)，不透明度小于等于0.01(alpha <= 0.01)，则UIView不接受事件处理。

UIScrollView
继承自UIView，拥有UI滚动功能。

UIControl
继承自UIView，拥有事件绑定和处理功能。

UITableView
继承自UIScrollView，通过读取数据创建UITableViewCell加入其中来进行数据展示，其中包含头部视图(Header View)，底部视图(Footer View)，多个 Sectio(每一个 Section 包含多个 Cell，一个Section 头部，一个 Section 底部)。其中有UITableViewDataSource接口成员弱指针，用于调用并且返回数据，实现表格的数据展示。有UITableViewDelegate接口成员弱指针，用于执行触发生命周期中的事件。

UIViewController
UIViewController是控制类，用于控制UIView等的逻辑功能，对应MVC的C。

UINavigationController
继承自UIViewController，提供UIView界面导航功能，作为一个父控制器来管理子控制器(控制器栈)，第一个入栈的控制器为根控制器(RootViewController)，提供对控制器的入栈出栈等操作，提供导航条(在UINavigationController中对UIViewController进行了类别定义)。