meili 美丽联合集团

启动性能优化实践

如何从 0 - 1

崆崆 #7 Mday

背景

What I Will Talk

What I Will Talk

定位问题

What I Will Talk

定位问题

解决问题

What I Will Talk

定位问题

解决问题

保持优化

定位

收集资料

收集资料

Google

Developer

opensource-apple

stackoverflow

收集资料

Google

Developer

opensource-apple

stackoverflow

WWDC

session 406 Optimizing App Startup Time

session 225_ up and running making a great impression with every launch

启动是什么

启动是什么

热启动

App and data already in memory

启动是什么

热启动

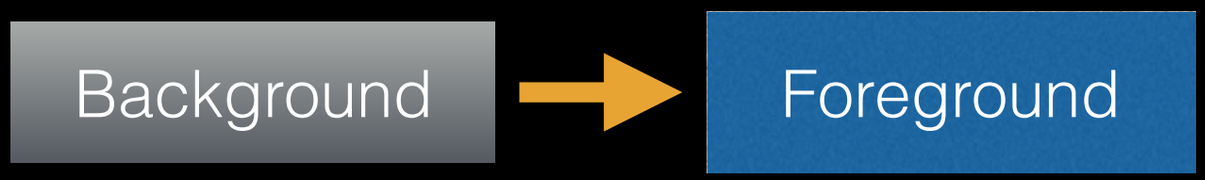
App and data already in memory

Background

启动是什么

热启动

App and data already in memory



启动是什么

热启动

App and data already in memory

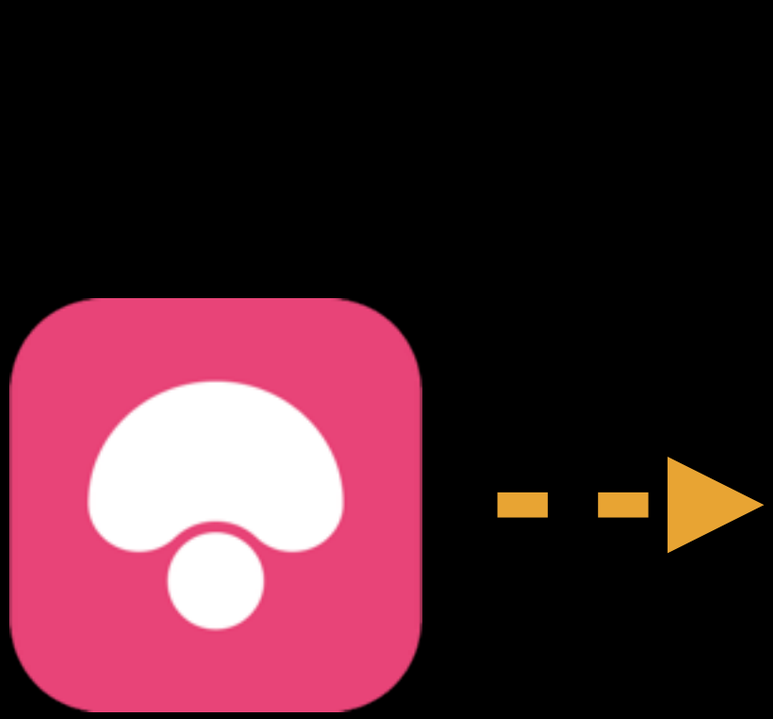
冷启动

App is not in kernel buffer cache

冷启动

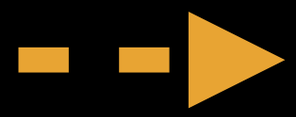
用户点击 icon 到 显示 非「LaunchImage」的过程

用户点击 icon 到 显示 非「LaunchImage」的过程



用户点击 icon 到 显示 非「LaunchImage」的过程

PreMain



AfterMain

PreMain

AfterMain

蘑菇街的摸索过程

蘑菇街的摸索过程

整理代码

蘑菇街的摸索过程

整理代码

埋点测量

蘑菇街的摸索过程

整理代码

埋点测量

排查遗漏

整理代码

App 基本参数设置

安全模块加载

网络库初始化

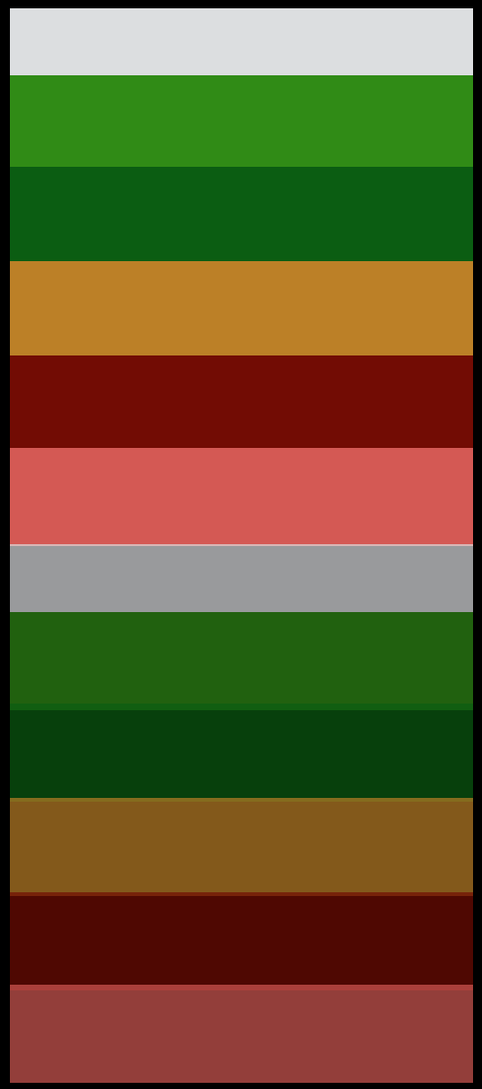
配置中心初始化

打点库初始化

基础路由初始化

页面白名单初始化

UI框架初始化



第三方库加载

User Agent设置

导航栏全局样式设定

图片库加载

HTTPDns库加载

用户管理模块加载

……

埋点测量

埋点测量

整个启动阶段的埋点

```
#pragma mark - main.m
int main(int argc, char *argv[])
{
    @autoreleasepool {
        [MGJLaunchPerfomance didBeginLaunch];
       return UIApplicationMain(argc, argv, nil,
NSStringFromClass([AppDelegate class]));
    }
}
```

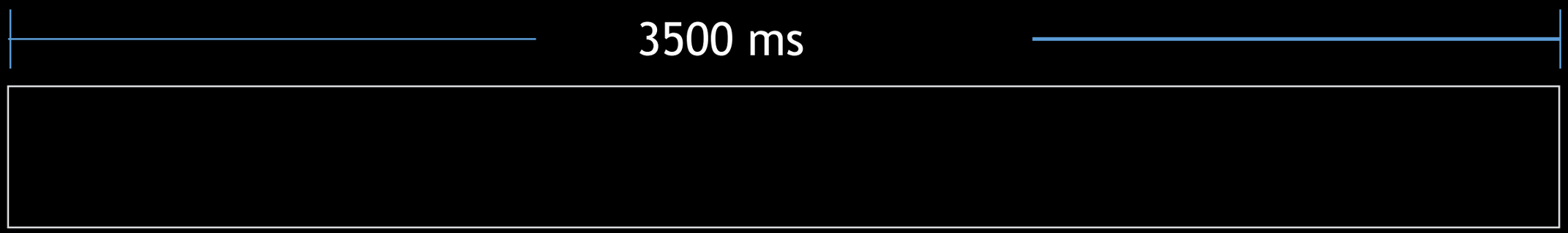
```
#pragma mark - AppDelegate
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    xxxxxx
    xxxxxx
    /// 一系列业务加载代码
    [MGJLaunchPerfomance didFinishLaunch];
    return YES;
}
```

```
#pragma mark - main.m
int main(int argc, char *argv[])
{
    @autoreleasepool {
        [MGJLaunchPerfomance didBeginLaunch];
        return UIApplicationMain(argc, argv, nil,
NSStringFromClass([AppDelegate class]));
    }
}
```

```
#pragma mark - AppDelegate
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    xxxxxx
    xxxxxx
    /// 一系列业务加载代码
    [MGJLaunchPerfomance didFinishLaunch];
    return YES;
}
```

埋点测量

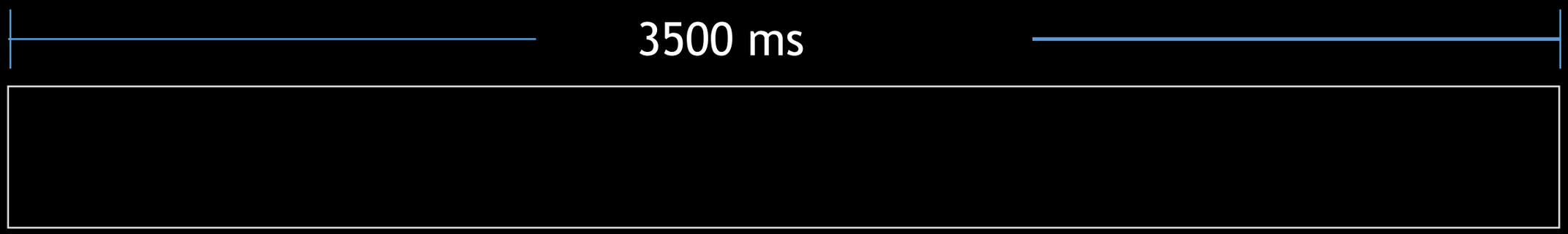
整个启动阶段的埋点



埋点测量

整个启动阶段的埋点

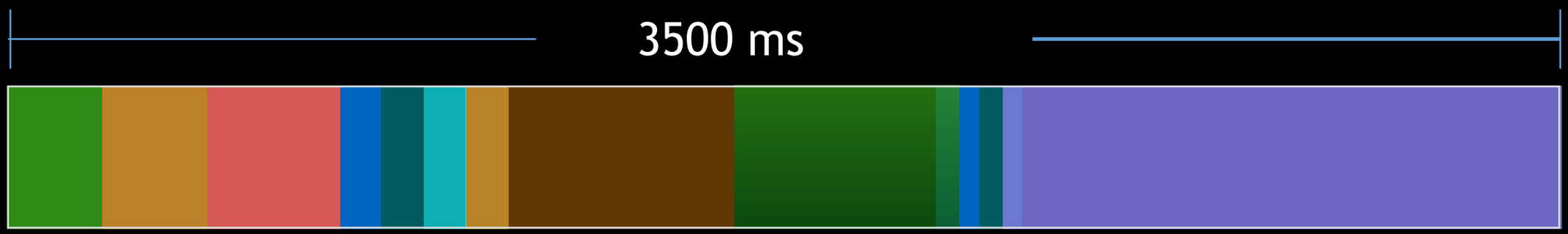
针对各个阶段业务埋点



埋点测量

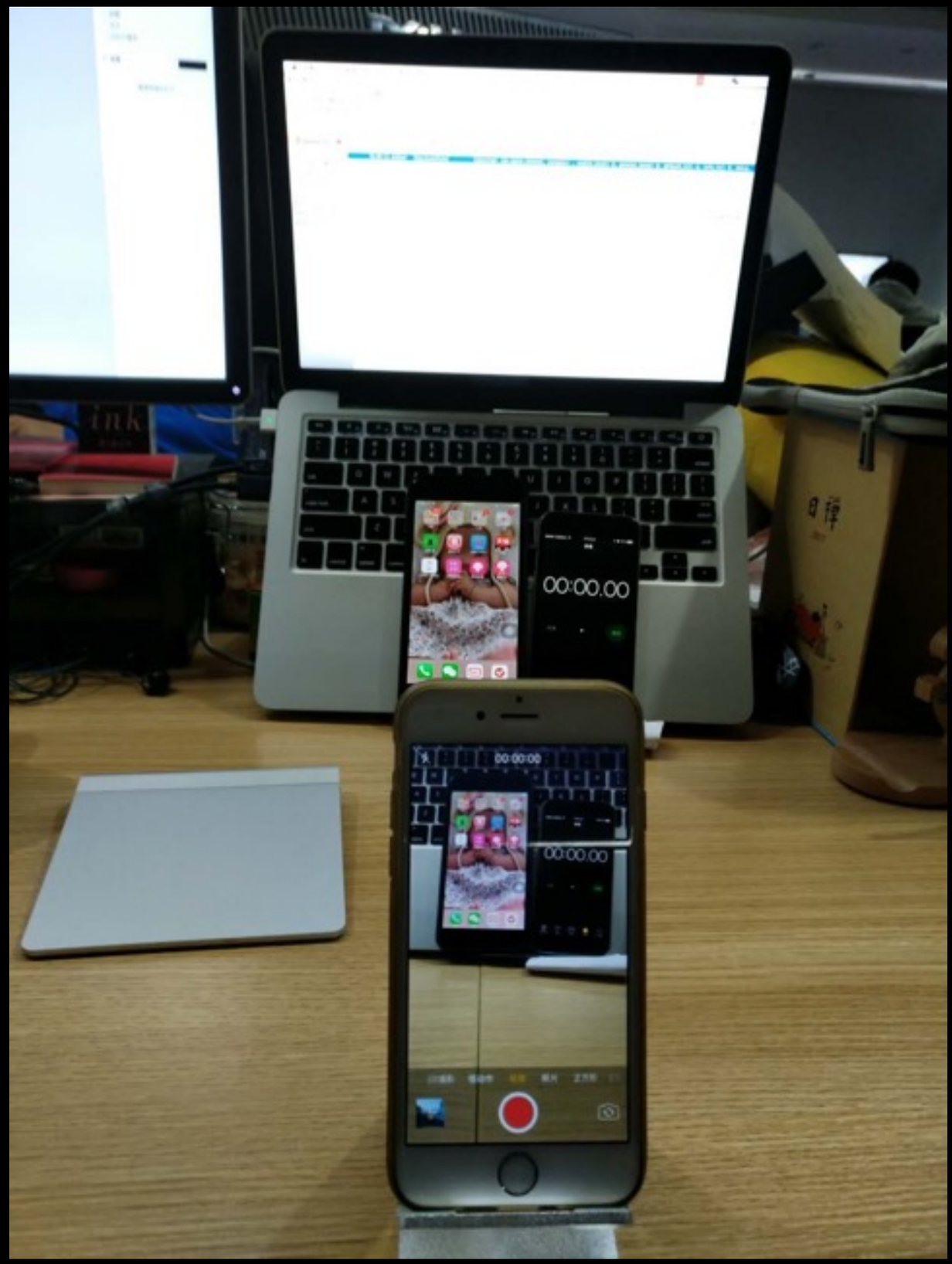
整个启动阶段的埋点

针对各个阶段业务埋点



我们启动 App 的
实际感受

不止 3.5 s


00:00.00
00:00:00
00:00.00

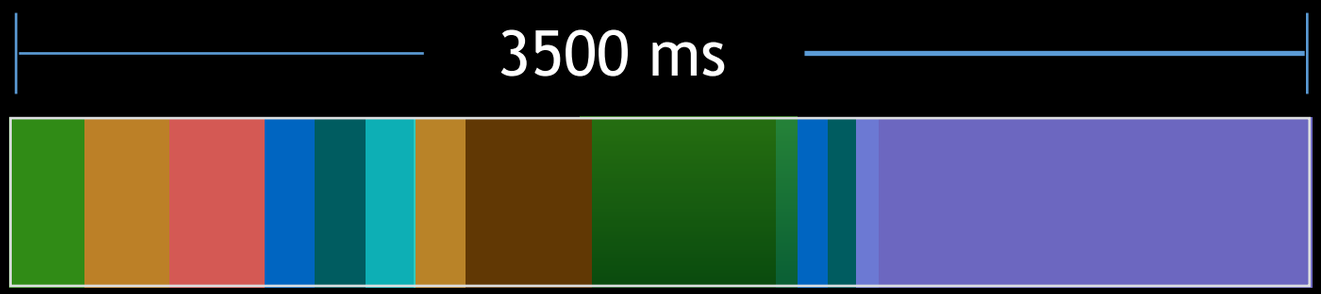

MOGU
STREET
meili 美丽联合集团


MOGU
STREET
meili 美丽联合集团


红人店
每日更新 流行好货随心购
新品
2月22日 上新
早春出街chic流苏上衣，让你少女心爆棚

实际耗时分布

非 UI 加载耗时：3500 ms

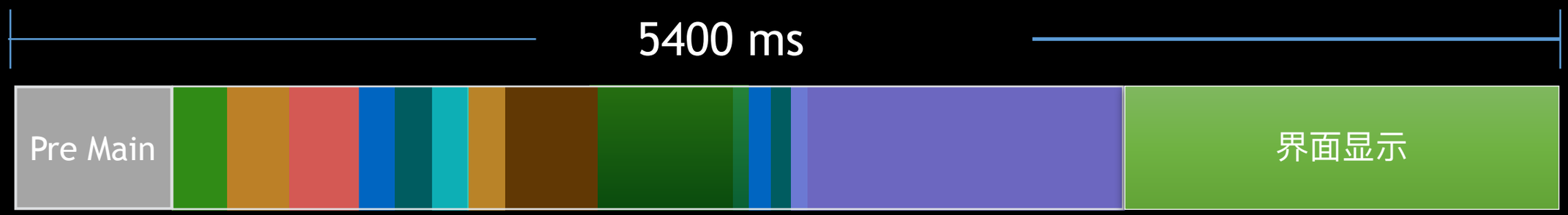


实际耗时分布

PreMain：400 ms

非 UI 加载耗时：3500 ms

UI 相关加载耗时：1500 ms



定位问题汇总

Pre Main加载

主线程串行执行

大文件 I/O 操作

第三方库加载

启动阶段 非必要的 View 的操作

页面显示逻辑及处理复杂且混乱

解 ◎ 決

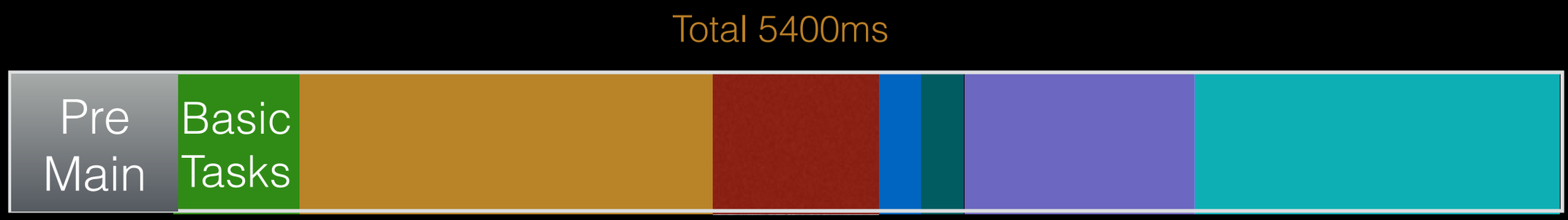
用户点击 icon 到 显示 非「LaunchImage」的过程

让用户更快的看到 非「LaunchImage」界面

针对问题解决方案

PreMain 优化　　按照 WWDC 2016 session 406 的指导操作

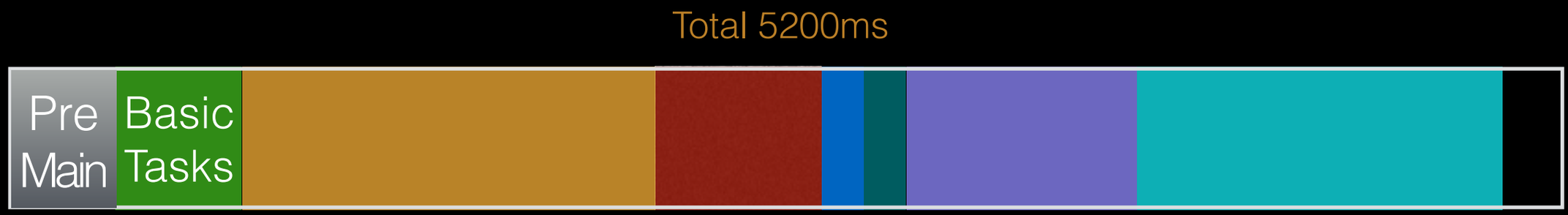
- 删除一些用不到的 framework
- 合并和删除一些 重复及不用的类
- 合并和删除一些 重复及不用的方法
- +load 方法 改成 懒加载或者 +initialize



针对问题解决方案

PreMain 优化　　按照 WWDC 2016 session 406 的指导操作

- 删除一些用不到的 framework
- 合并和删除一些 重复及不用的类
- 合并和删除一些 重复及不用的方法
- +load 方法 改成 懒加载或者 +initialize



针对问题解决方案

PreMain 优化

按照 WWDC 2016 session 406 的指导操作

主线程串行执行

延后执行或者异步并行执行

针对问题解决方案

PreMain 优化

按照 WWDC 2016 session 406 的指导操作

主线程串行执行

延后执行或者异步并行执行

大文件 I/O 操作

懒加载或者 拆分启动必须内容

针对问题解决方案

PreMain 优化	按照 WWDC 2016 session 406 的指导操作
主线程串行执行	延后执行或者异步并行执行
大文件 I/O 操作	懒加载或者 拆分启动必须内容
第三方库加载	延后执行或者异步并行执行

针对问题解决方案

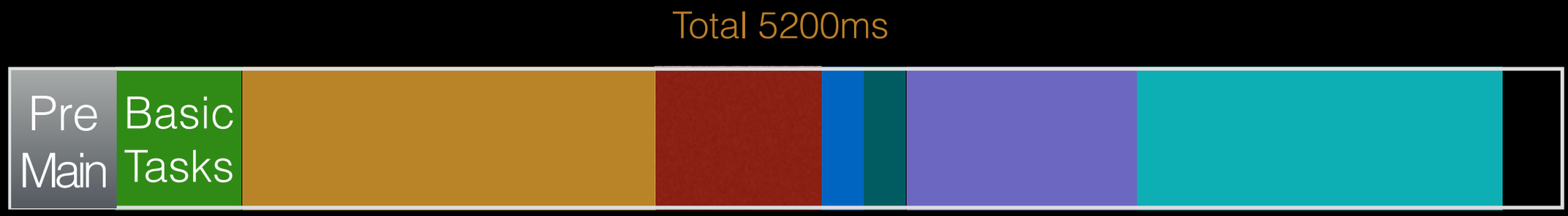
PreMain 优化	按照 WWDC 2016 session 406 的指导操作
主线程串行执行	延后执行或者异步并行执行
大文件 I/O 操作	懒加载或者 拆分启动必须内容
第三方库加载	延后执行或者异步并行执行
启动阶段 非必要的 View 的操作	懒加载或者VC生命周期管理

针对问题解决方案

PreMain 优化	按照 WWDC 2016 session 406 的指导操作
主线程串行执行	延后执行或者异步并行执行
大文件 I/O 操作	懒加载或者 拆分启动必须内容
第三方库加载	延后执行或者异步并行执行
启动阶段 非必要的 View 的操作	懒加载或者VC生命周期管理
页面显示逻辑及处理复杂且混乱	简化流程，清理显示代码

主要手段

- 异步化、延迟加载及懒加载
- 分拆 I/O
- 界面显示优化



- 异步及延迟加载

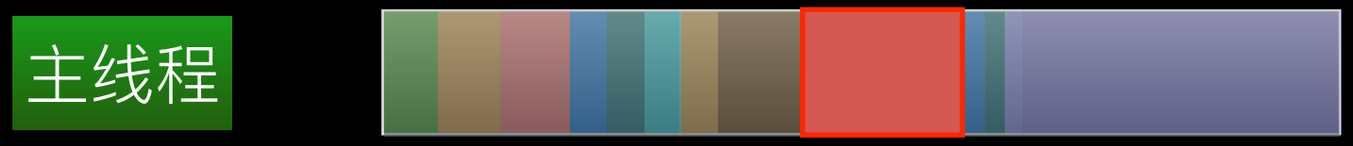
第三方库的异步加载

第三方库的异步加载

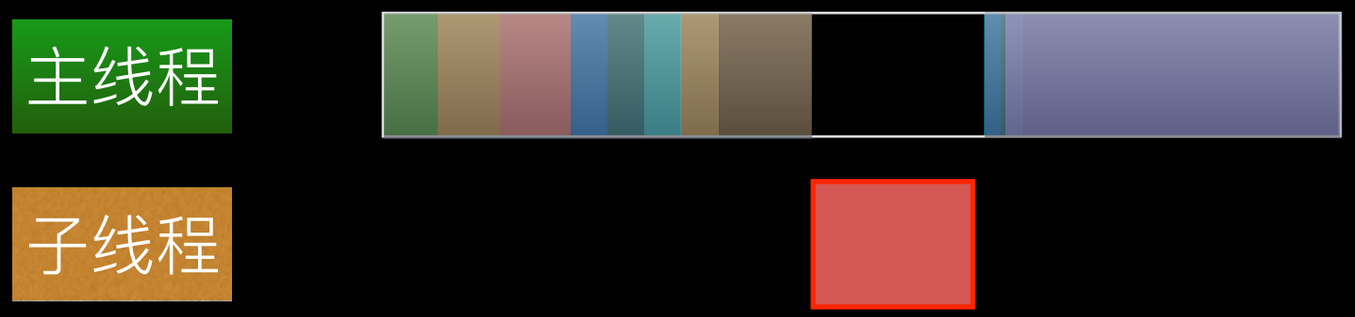
原状态：

第三方库的异步加载

原状态：



想象中：



第三方库的异步加载

原状态：

主线程

想象中：

主线程

子线程

实际情况：

主线程

子线程

第三方库的异步加载

URL
or
第三方

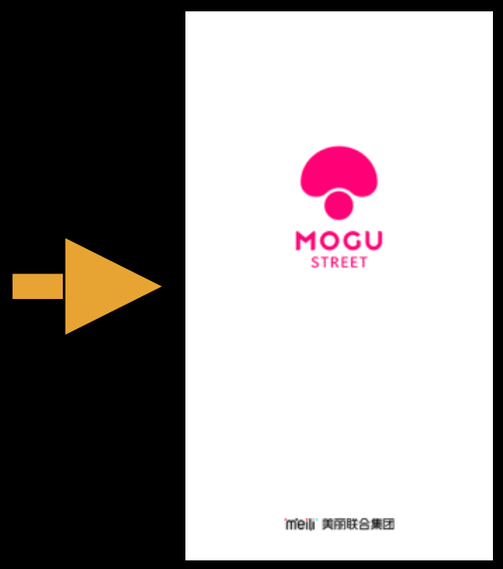
实际情况：

主线程

子线程

第三方库的异步加载

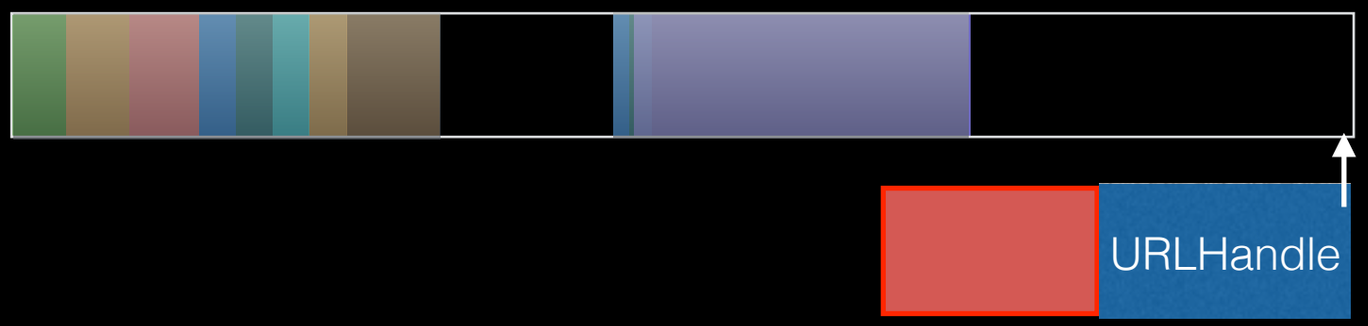
URL
or
第三方



实际情况：

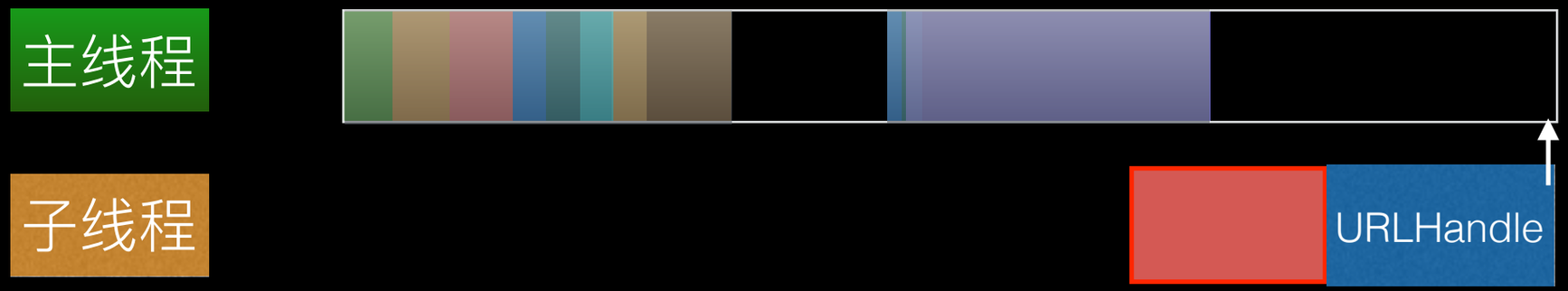
主线程

子线程

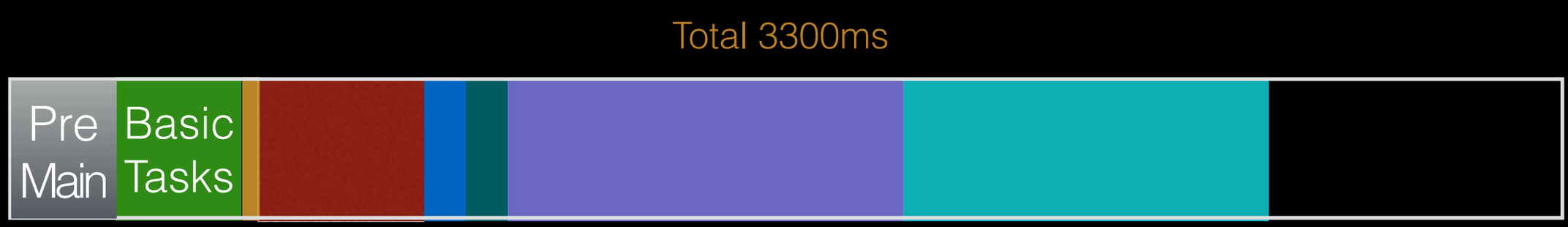


第三方库的异步加载

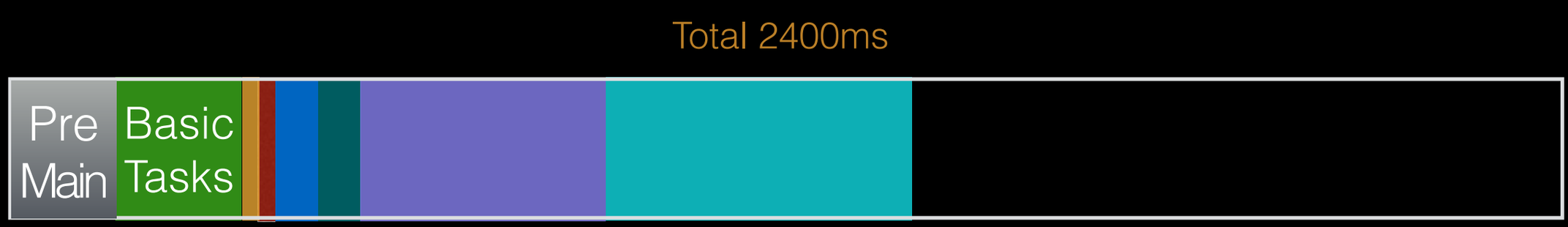
实际情况：



- I/O 分拆及优化

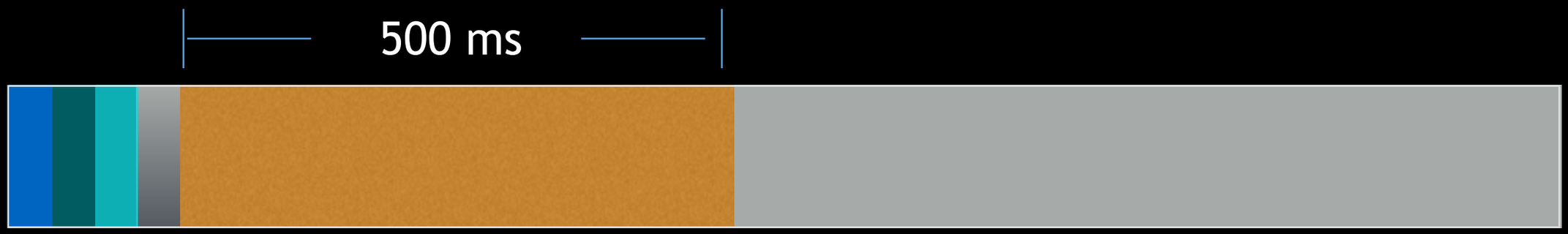


- I/O 分拆及优化

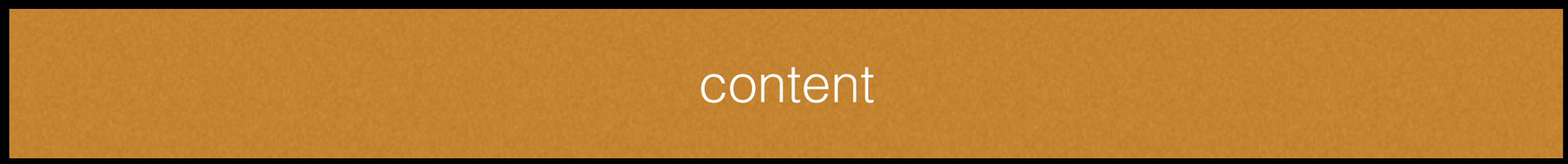


安卓读取渠道号

耗时

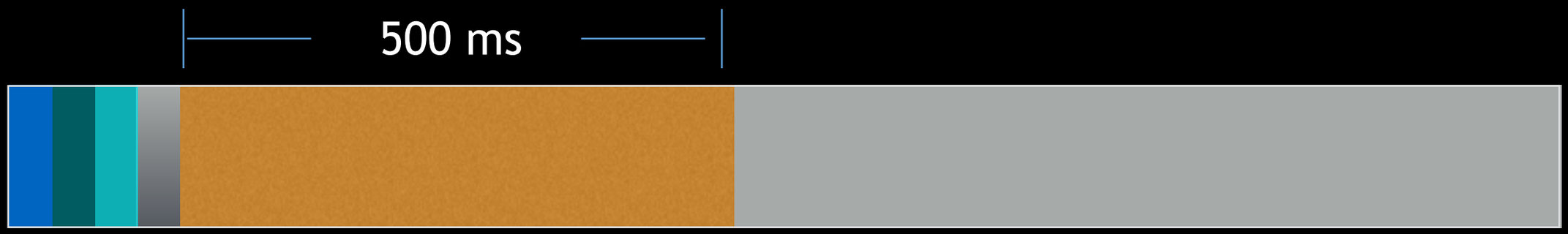


文件

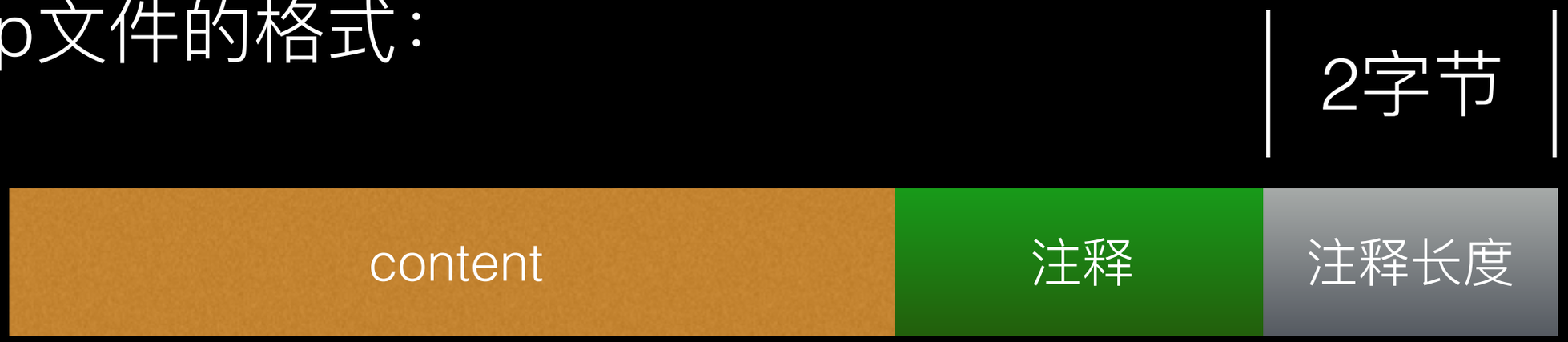


安卓读取渠道号

耗时

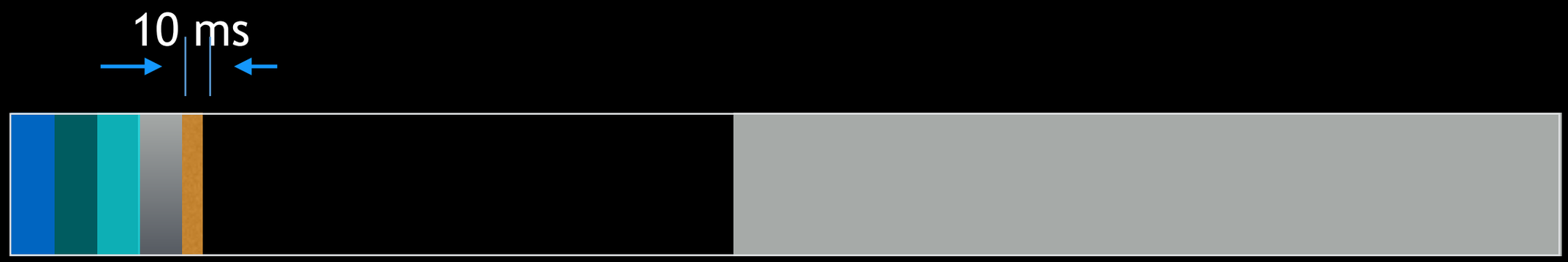


zip文件的格式：

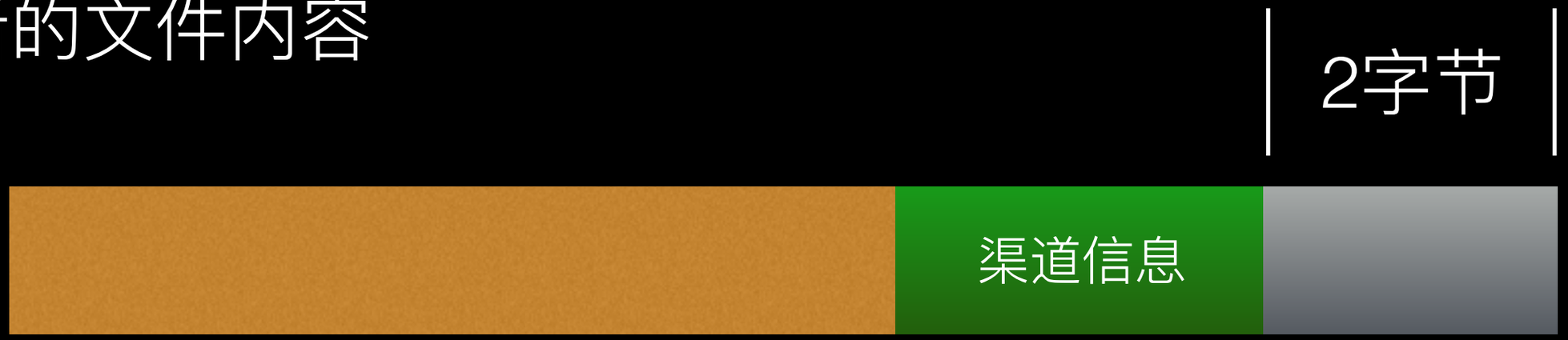


安卓读取渠道号

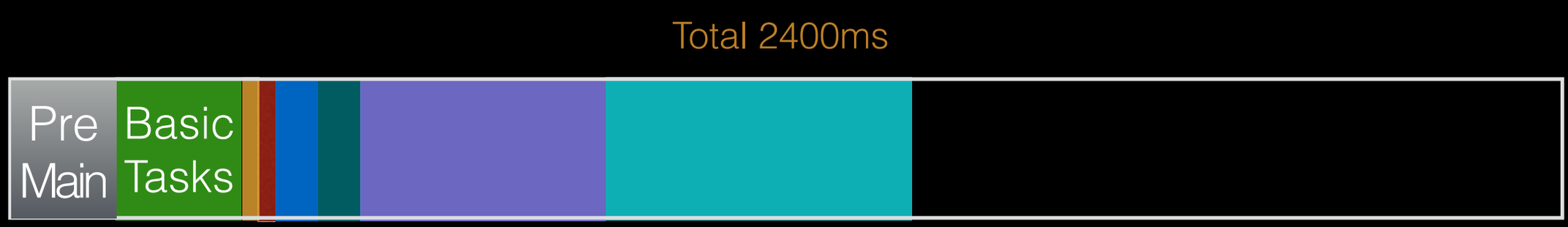
耗时



新的文件内容

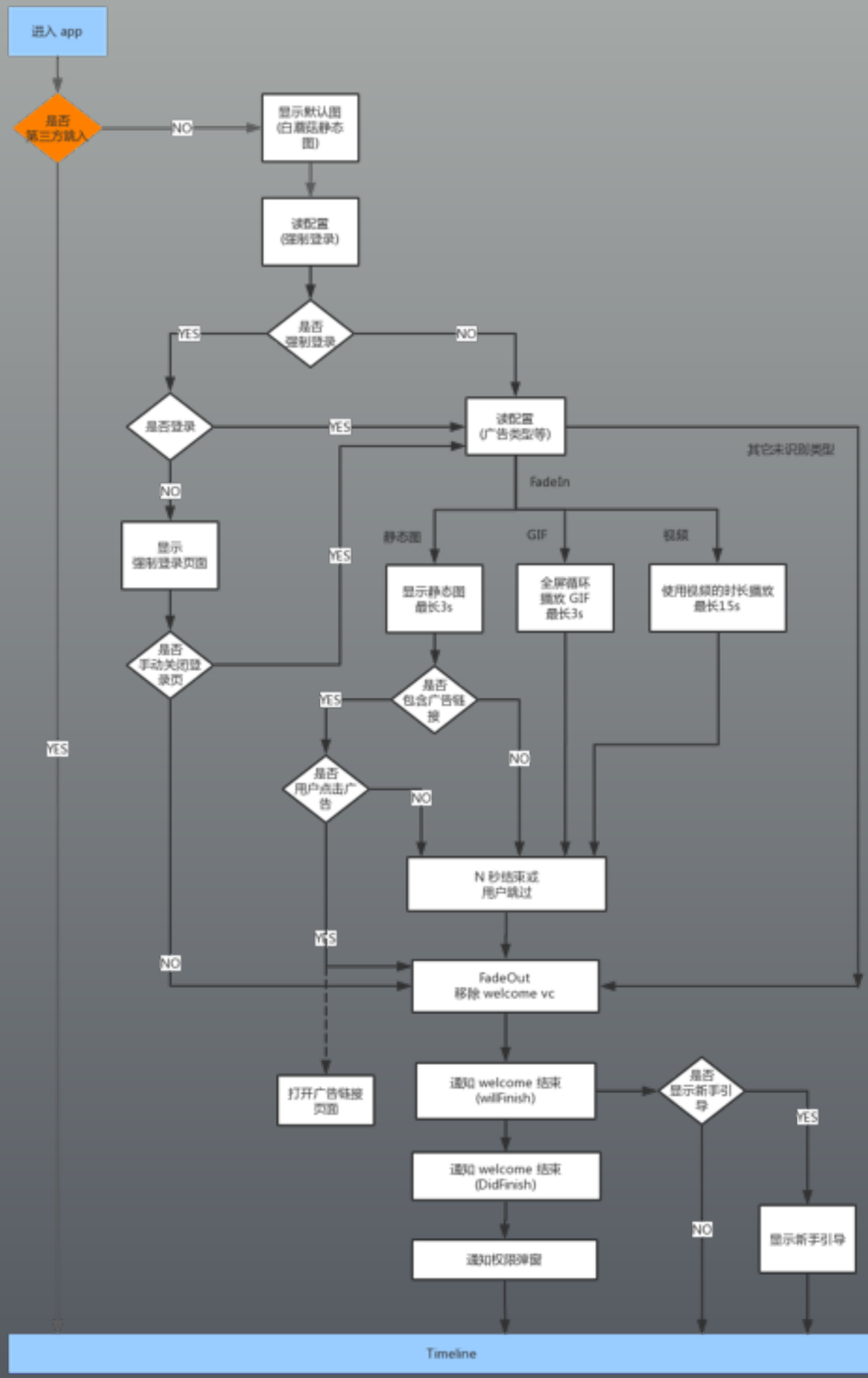


界面显示优化



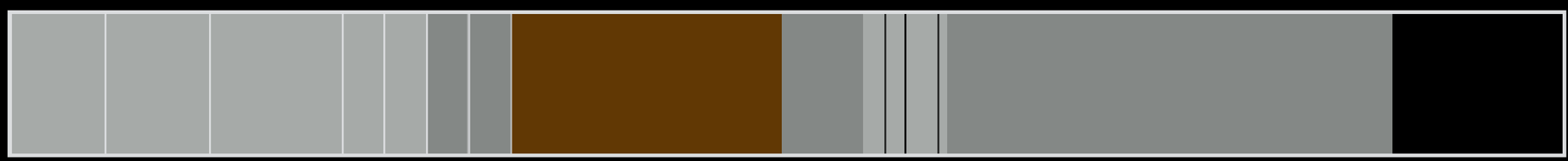
界面显示优化

Welcome 显示优化


进入 app
是否
第三方跳入
NO
显示默认图
(白底品静态
图)
读配置
(强制登录)
是否
强制登录
YES
NO
是否登录
YES
读配置
(广告类型等)
其它未识别类型
NO
FadeIn
显示
强制登录页面
静态图
GIF
视频
YES
显示静态图
最长3s
全屏循环
播放 GIF
最长3s
使用视频的时长播放
最长15s
是否
手动关闭登
录页
是否
包含广告链
接
YES
YES
NO
是否
用户点击广
告
NO
N 秒结束或
用户跳过
YES
NO
FadeOut
移除 welcome vc
打开广告链接
页面
通知 welcome 结束
(willFinish)
是否
显示新手引
导
YES
通知 welcome 结束
(DidFinish)
NO
显示新手引导
通知权限弹窗
Timeline

Welcome 显示优化

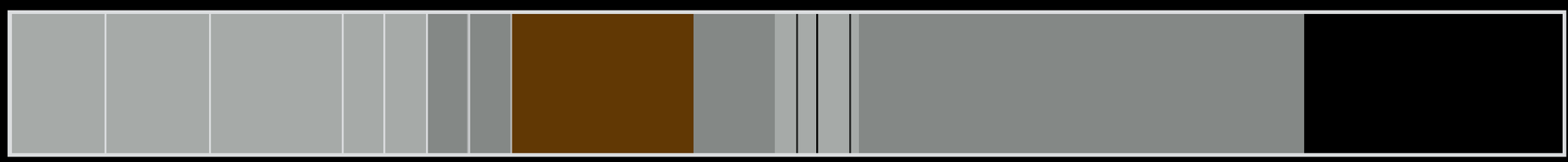
- 多余 view 的创建及数据加载



Welcome 显示优化

- 多余 view 的创建及数据加载

- 所有显示操作，不要使用 performSelector:afterDealy 之类的方法

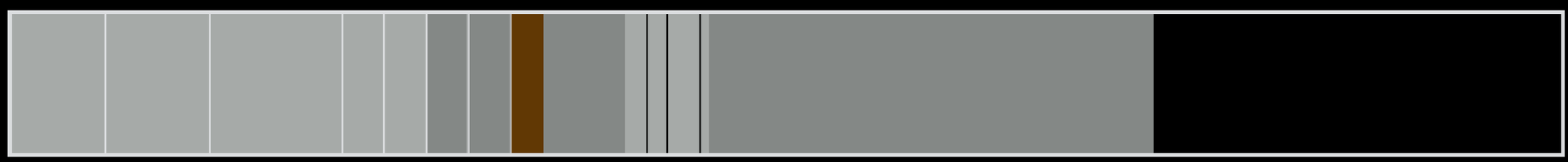


Welcome 显示优化

- 多余 view 的创建及数据加载

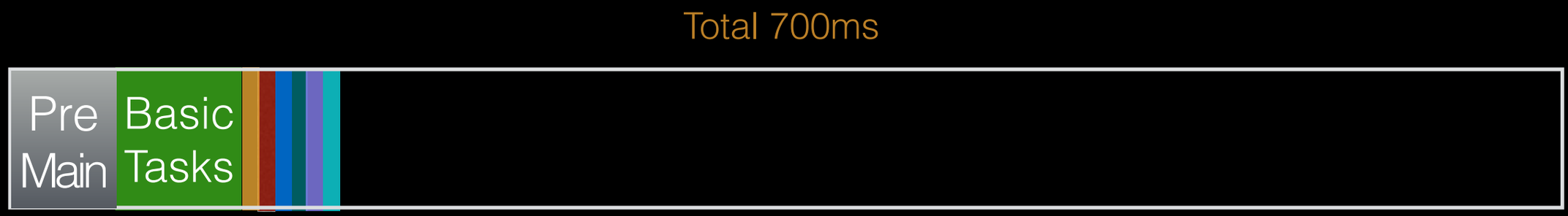
- 所有显示操作，不要使用 performSelector:afterDealy 之类的方法

- 界面显示的入口在 viewWillAppear，而非 viewDidAppear中



解决问题的方法

- 异步化、延迟加载及懒加载
- 分拆 I/O
- 界面显示优化



保持

启动任务管理器

- 业务无关

- 业务无关

- 根据业务场景分成 四个 队列

- 业务无关

- 根据业务场景分成 四个 队列

- 每个队列中任务，顺序执行

- 业务无关

- 根据业务场景分成 四个 队列

- 每个队列中任务，顺序执行

- 快速便捷的依赖管理

启动器结构图

启动任务

Task1 Task2 ..... TaskN

任务组装层

BasicAdapter

AppStartupAdapter

启动器

Dependency Checker

Queue Queue Queue Queue

QueueManager

启动器队列

- 基础任务队列

启动器队列

- 基础任务队列
- 主线程业务队列

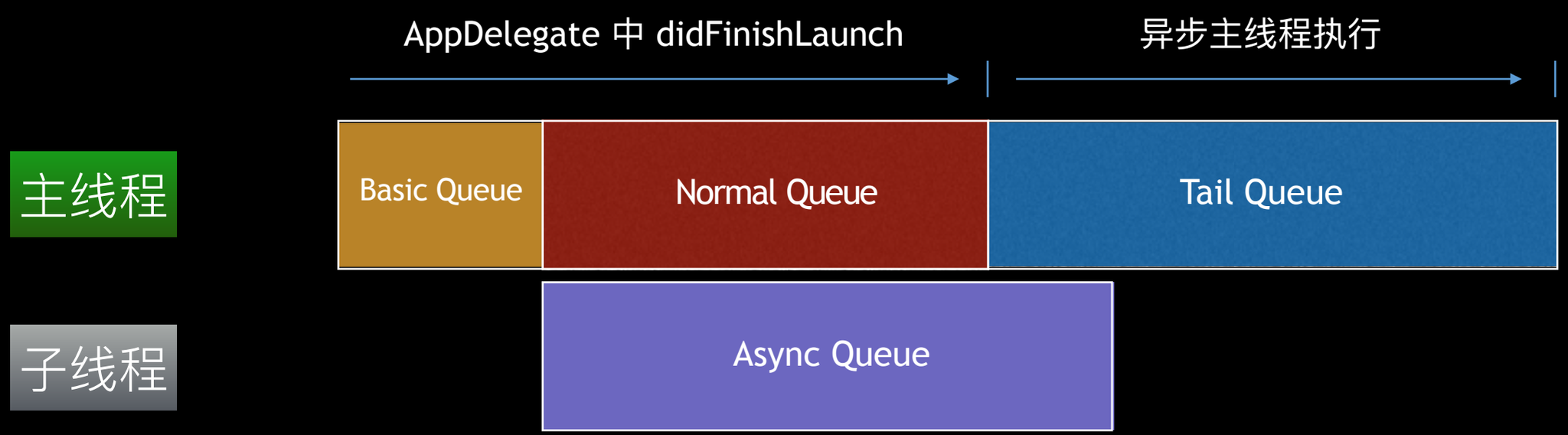
启动器队列

- 基础任务队列
- 主线程业务队列
- 子线程异步队列

启动器队列

- 基础任务队列
- 主线程业务队列
- 子线程异步队列

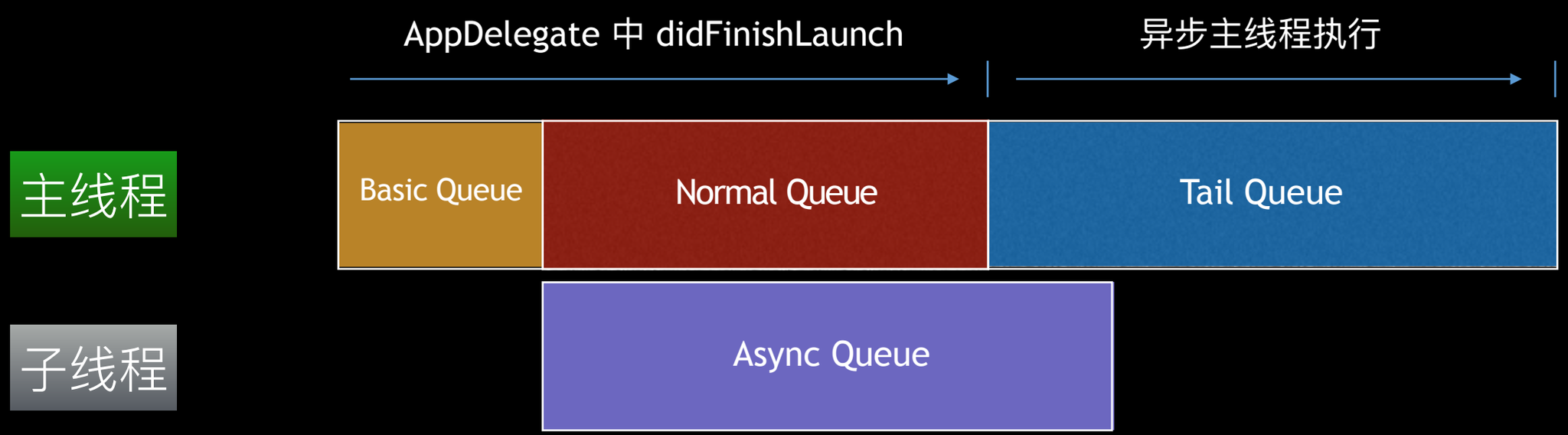
主线程异步队列



启动器队列

- 基础任务队列
- 主线程业务队列
- 子线程异步队列

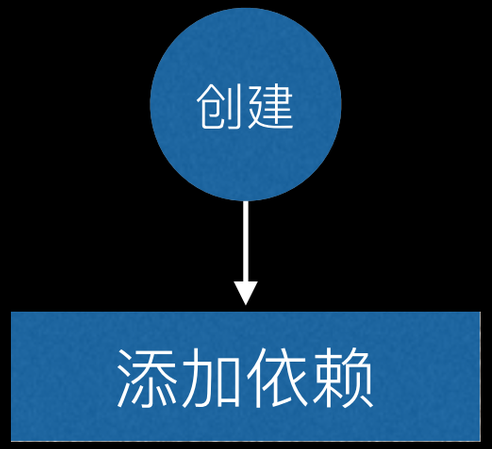
主线程异步队列



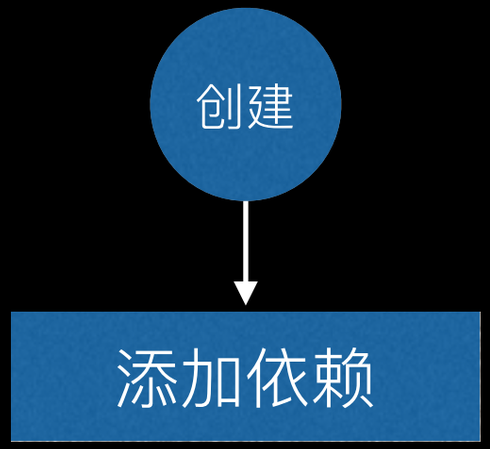
依赖管理

- 简单快速
- 区分环境

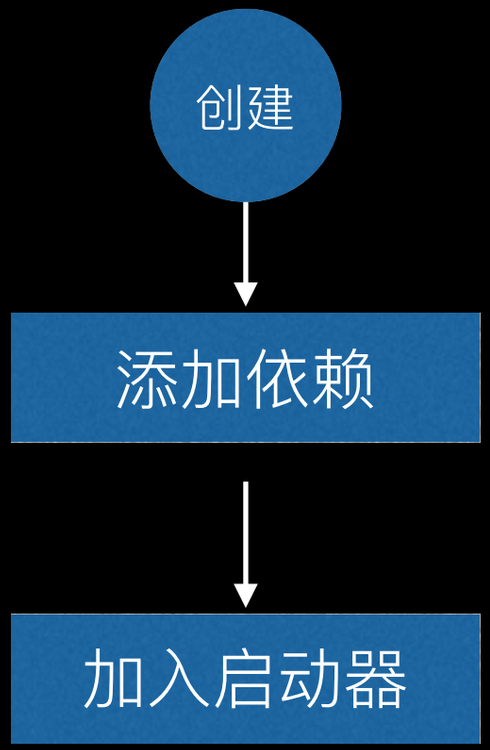
创建

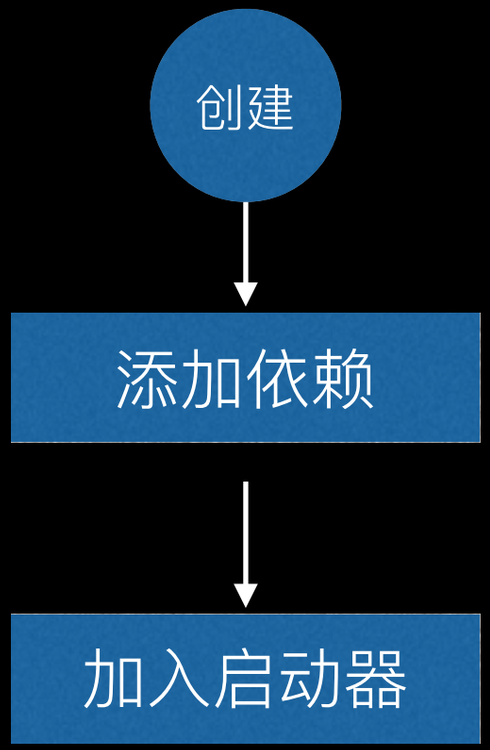



```
#pragma mark – HTTPSTask
@implementation HTTPSTask
- (void)init {
    xxxxxx
    [self addDependencyList:
      @[@“HoustonTask",
       @"MGJRequestTask"]];
    xxxxxx
}
```

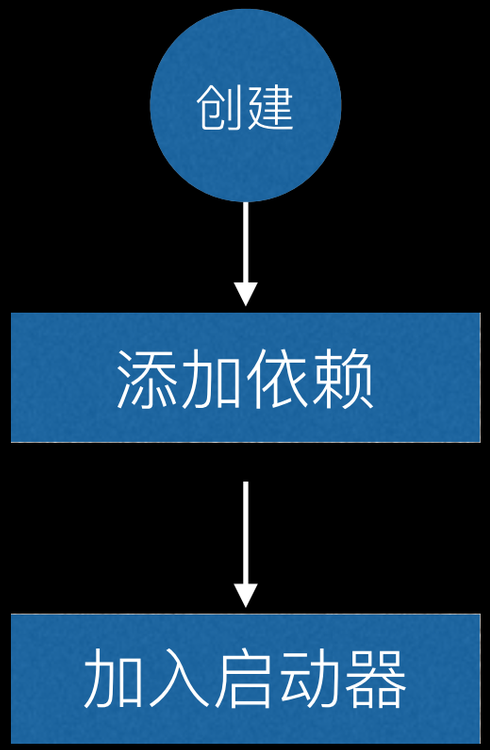



```
#pragma mark – HTTPSTask
@implementation HTTPSTask
- (void)init {
    xxxxxx
    [self addDependencyList:
      @[@“HoustonTask",
       @"MGJRequestTask"]];
    xxxxxx
}
```

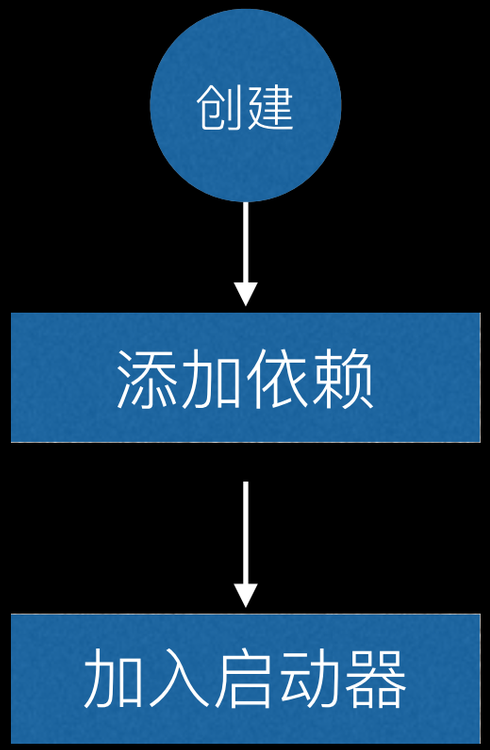

创建
添加依赖
加入启动器



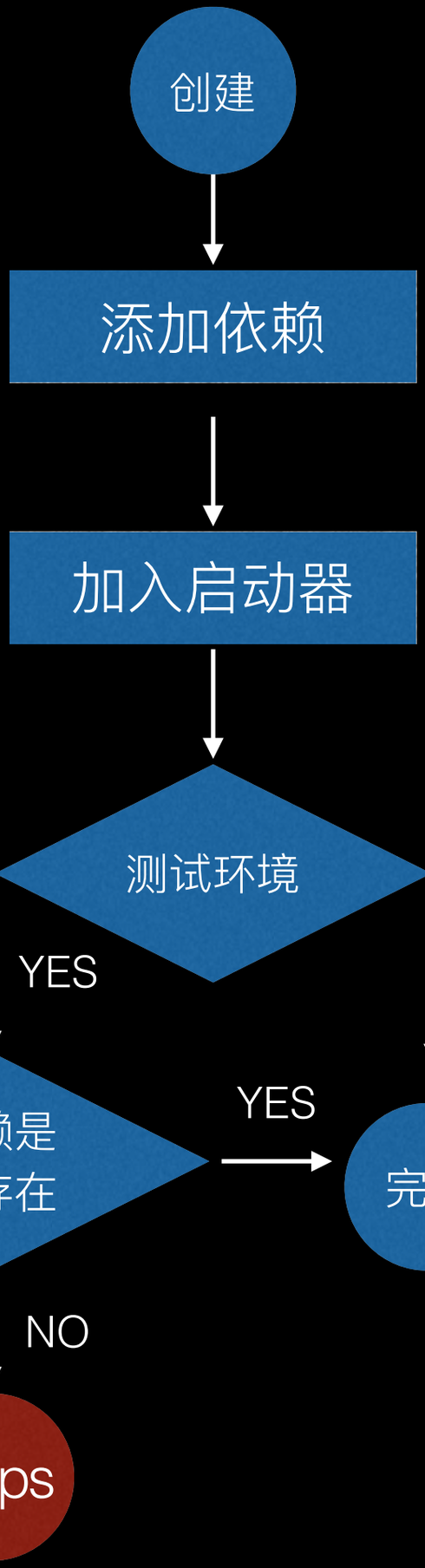

```objc
#pragma mark - 启动器添加任务校验
- (void)checkTaskDependencyWithTask:(id <ALCTaskProtocol>)task {
    if ([self envForTesting]) {
        [task.dependencyList enumerateObjectsUsingBlock:^(NSString *dependencyTask,
NSUInteger idx, BOOL *stop) {
            if (![self.taskNameMap objectForKey:dependencyTask])
                @throw [NSException exceptionWithName:AppInitializeExceptionName
reason:[NSString stringWithFormat:@"%@ dependency not load", task.taskName]
userInfo:nil];
        }];
    }
}
```




```objc
#pragma mark - 启动器添加任务校验
- (void)checkTaskDependencyWithTask:(id <ALCTaskProtocol>)task {
    if ([self envForTesting]) {
        [task.dependencyList enumerateObjectsUsingBlock:^(NSString *dependencyTask,
NSUInteger idx, BOOL *stop) {
            if (![self.taskNameMap objectForKey:dependencyTask])
                @throw [NSException exceptionWithName:AppInitializeExceptionName
reason:[NSString stringWithFormat:@"%@ dependency not load", task.taskName]
userInfo:nil];
        }];
    }
}
```




```
#pragma mark - 启动器添加任务校验
- (void)checkTaskDependencyWithTask:(id <ALCTaskProtocol>)task {
    if ([self envForTesting]) {
        [task.dependencyList enumerateObjectsUsingBlock:^(NSString *dependencyTask,
NSUInteger idx, BOOL *stop) {
            if (![self.taskNameMap objectForKey:dependencyTask])
                @throw [ALCException exceptionWithName:task.taskName];
        }];
    }
}
```


创建
添加依赖
加入启动器

测试环境
YES
NO
依赖是
否存在
YES
完成
NO
Opps

总结

「总结」

- 一个或多个工具帮助定位问题

如果没有，自己做一个

- 分阶段并可量化的目标

有了目标，才能确立方案及步骤

- 可靠的数据

有数据，才能持续改进及发现问题

- 永远要质疑之前的结论

方案永远不是「最佳」

Q & A

'meili' 美丽联合集团

THANK YOU

崆峒 #7 Mday