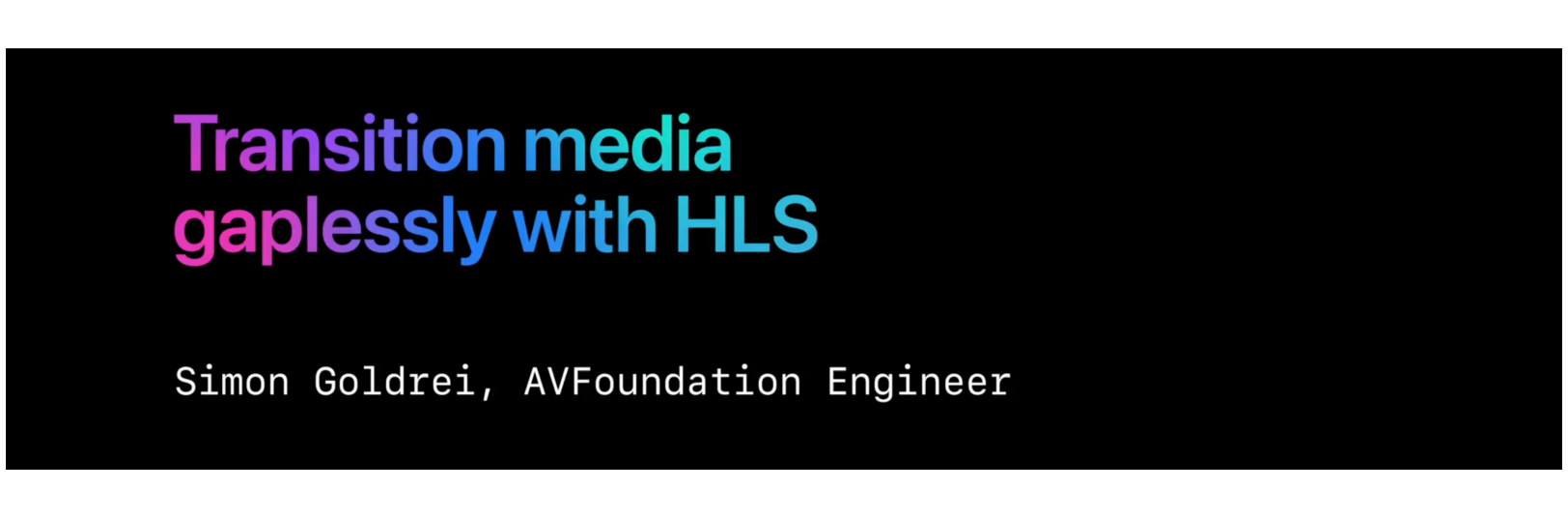
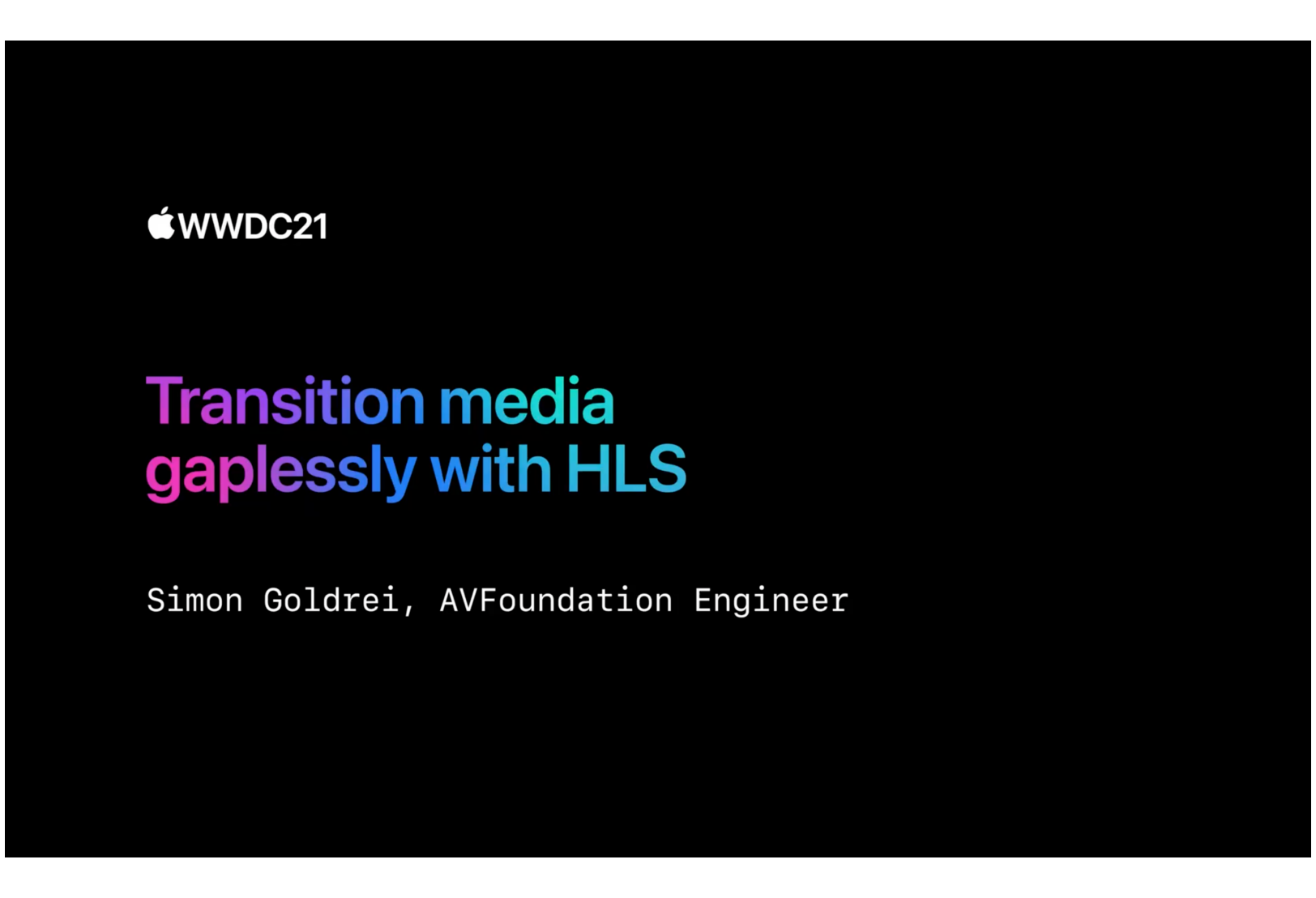


【WWDC21 10142】使用 HLS 让媒体无缝切换

老司机技术周报 已关注
预计阅读时间13分钟 7 月前

> 作者：fakeGourmet，目前在字节跳动从事 iOS 开发，对音视频领域感兴趣。
>
> 审核：jojotov, iOS 开发，SwiftGG 翻译组成员，目前任职字节跳动，负责抖音直播客户端相关工作。



Overview

本文基于 Session 10142 梳理。

本文将向您介绍 HLS 无缝切换功能的好处、条件以及使用方式，最后还将介绍一个资源切割的小技巧。

本篇

本 session 介绍的是 HLS 无缝切换的新特性，由苹果流媒体团队的工程师 Simon Goldrei 讲解。

> HTTP Live Streaming (HLS) 是苹果公司在 2009 年推出的基于 HTTP 的流媒体协议。因为 HLS 是基于 HTTP 协议的流媒体传输协议，所以 HLS 可以复用任何 HTTP 相关的基础设施。相对于 RTP 来说，HLS 被防火墙拦截的概率会更小（因为大部分防火墙都允许 HTTP 数据通过），也更容易使用 CDN 来传输媒体流，因此 HLS 得到了广泛的应用。
>
> 注意，在阅读本文前我们假设你已经对 HLS 技术有所了解。如果你还不是很了解，建议先阅读 iOS 边学边记 HLS 协议 m3u8 ts 详解。

如果你想了解其他有趣的 HLS 新特性，请参考：

- WWDC21 10141 - 使用 HLS Content Steering 提升全球范围流媒体的可用性
- WWDC21 10143 - 在 AVFoundation 中探索 HLS 变体

不够丝滑的 AVQueuePlayer

我们通常在需要播放一段视频时使用 AVPlayer。而 AVPlayer 一次只能播放一个视频文件。对于一次性播放多个视频的情况，苹果推出了 AVQueuePlayer 供开发者使用。

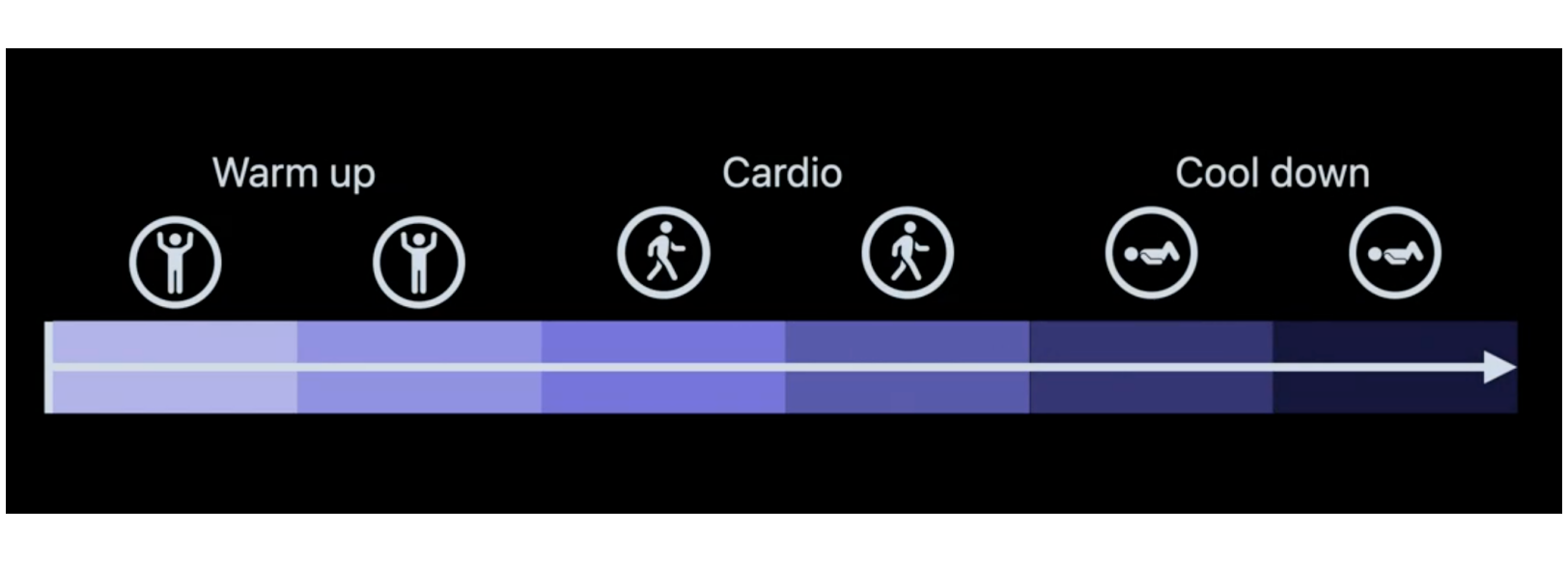
> AVQueuePlayer 是 AVPlayer 的一个子类，它在 AVPlayer 的基础上内部维护了一个播放列表，并对外提供了一系列 API 去播放这个列表。具体的 API 可以浏览 官方文档。

虽然 AVQueuePlayer 提供了播放多个媒体的能力，但在媒体切换的过渡点上还存在一些瑕疵。特别是播放 HLS 媒体的时候，会存在声音毛刺、短暂的画面卡顿甚至是一个不知道何时才能完成的缓冲条等现象。

无缝切换带来的好处

而本篇 session 便来介绍如何解决这个问题，让媒体之间能够无瑕疵地进行切换，让用户体验丝滑般的享受。无缝切换能带来许多好处：

- 对于专辑类媒体能提供一个符合用户长久以来习惯的无缝体验
- 线性编码（这个没有具体说明是啥 >_<，笔者个人理解是能直接通过 AVQueuePlayer 设定播放顺序？）
- 提供一个动态的、交互式场景的体验。 这里举个跑步健身的例子，你可以设定三个步骤，热身、有氧运动、冷却。对每个步骤分别设定一首 BGM，比如热身的时候设定「运动员进行曲」，运动时来首激情的摇滚，运动完听一曲经典萨克斯「Going Home」，回忆回忆当年夕阳西下踢完足球放学回家的滋味~



无缝切换的条件

前面说的这么好。那么，如何才能让 AVQueuePlayer 支持无缝切换呢？是让用户对着屏幕喊一声“芝麻开门”吗？当然不是。要支持无缝切换，需要达到一定的条件才行。

你的 HLS 播放列表需要满足以下几个条件：

- 需要使用苹果的 Fairplay Streaming 技术

> FairPlay 是苹果公司研发的一种数字版权管理 （DRM） 技术。它将加密的 AAC 音频层内置于 MP4 多媒体文件格式中，以保护通过 iTunes Store 发售作品的版权，只允许经过授权的设备才能播放内容。
>
> 关于 Fairplay Streaming 技术可以观看 WWDC20-10655 了解详尽信息。

- 需要在 `EXT-X-STREAM-INF` 标签的 `CODECS` 属性中指定编解码器
- 需要在 `EXT-X-MEDIA` 标签的 `CHANNELS` 属性中指定频道数量
- 变体之间的配置需要保持一致。比如两段音频之间的采样率（Sample Rate）和位深（Bits Depth）需要保持一致，否则无缝切换会失效
- 需要遵守 CMAF 标准，同时 Simon 还建议使用 elst（Edit List Box） 来表示起始帧和剩余帧的存在

> CMAF（Common Media Application Format） 是一个可扩展的标准，用于分段媒体的编码和打包，以便自适应地分发到用户终端设备上进行解码。CMAF 抽象出了一个应用模型，该模型存在多个实现，包括 HLS 和 MPEG DASH。
>
> elst 全称 Edit List Box，mp4 文件中不一定都含有这个 box。该 box 作用是使某个 track 的时间戳产生偏移。苹果在 iOS 13.1 以后支持使用 elst，详情见 官方文档。关于 elst 的内容可以参考 mp4 文件 elst 研究。

内容提供方在遵守了以上的条件之后就能进行无缝的切换了。下面举个例子。

> 注意，这个例子中包含媒体组（Media Group）相关技术，细节请参考 官方文档。

```
#EXTM3U

#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac-lc",CHANNELS="2",URI="./AAC_LC/prog_index.m3u8"
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="he-aac",CHANNELS="2",URI="./HE_AAC/prog_index.m3u8"

#EXT-X_STREAM-INF:BANDWIDTH=3900000,CODECS="avc1.64001f,mp4a.40.5",AUDIO="he-aac"
./720P/prog_index.m3u8

#EXT-X_STREAM-INF:BANDWIDTH=4000000,CODECS="avc1.64001f,mp4a.40.2",AUDIO="aac-lc"
./720P/prog_index.m3u8

#EXT-X_STREAM-INF:BANDWIDTH=8900000,CODECS="avc1.640028,mp4a.40.5",AUDIO="he-aac"
./1080P/prog_index.m3u8

#EXT-X_STREAM-INF:BANDWIDTH=9000000,CODECS="avc1.640028,mp4a.40.2",AUDIO="aac-lc"
./1080P/prog_index.m3u8
```

这个播放列表中有四个变体。其中一对是 720p，另外一对是 1080p。每一对都分别设定了两个不同的编解码器，定义为 he-aac 和 aac-lc。

假设当前在 he-aac 和 720p 条件下播放，那么播放下一段媒体时，AVQueuePlayer 将会选择同样是 he-aac 和 720p 的变体来播放。

网络的好坏不会影响这个选择。即便在播放下一段前一刻网络速度爆表，也不会切换到更高清的变体上去。当然，如果切换完成后网络条件还是很好，那么此时播放器会切换到更高清的变体上去。

那么如果下一个 m3u8 主播放列表中没有提供 he-aac 的变体怎么办？下面就是一个只有 aac-lc 编解码器的例子。

```
#EXTM3U

#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac-lc",CHANNELS="2",URI="./AAC_LC/prog_index.m3u8"
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="he-aac",CHANNELS="2",URI="./HE_AAC/prog_index.m3u8"

#EXT-X_STREAM-INF:BANDWIDTH=4000000,CODECS="avc1.64001f,mp4a.40.2",AUDIO="aac-lc"
./720P/prog_index.m3u8

#EXT-X_STREAM-INF:BANDWIDTH=9000000,CODECS="avc1.640028,mp4a.40.2",AUDIO="aac-lc"
./1080P/prog_index.m3u8
```

在这个播放列表中，播放器无法找到一个条件一致的变体（即带 he-aac 编解码器的变体），无缝切换也就失效了。此时播放器会自己选择一个合适的变体进行切换。

代码示例

Simon 提供了一个 AVQueuePlayer 的简单示例。

```
let item1 = AVPlayerItem(url: url1)
let item2 = AVPlayerItem(url: url2)

let player = AVQueuePlayer()

player.insert(item1, after: nil)
// 设置第二段视频紧接着第一段播放
player.insert(item2, after: item1)

player.play()
```

示例效果：

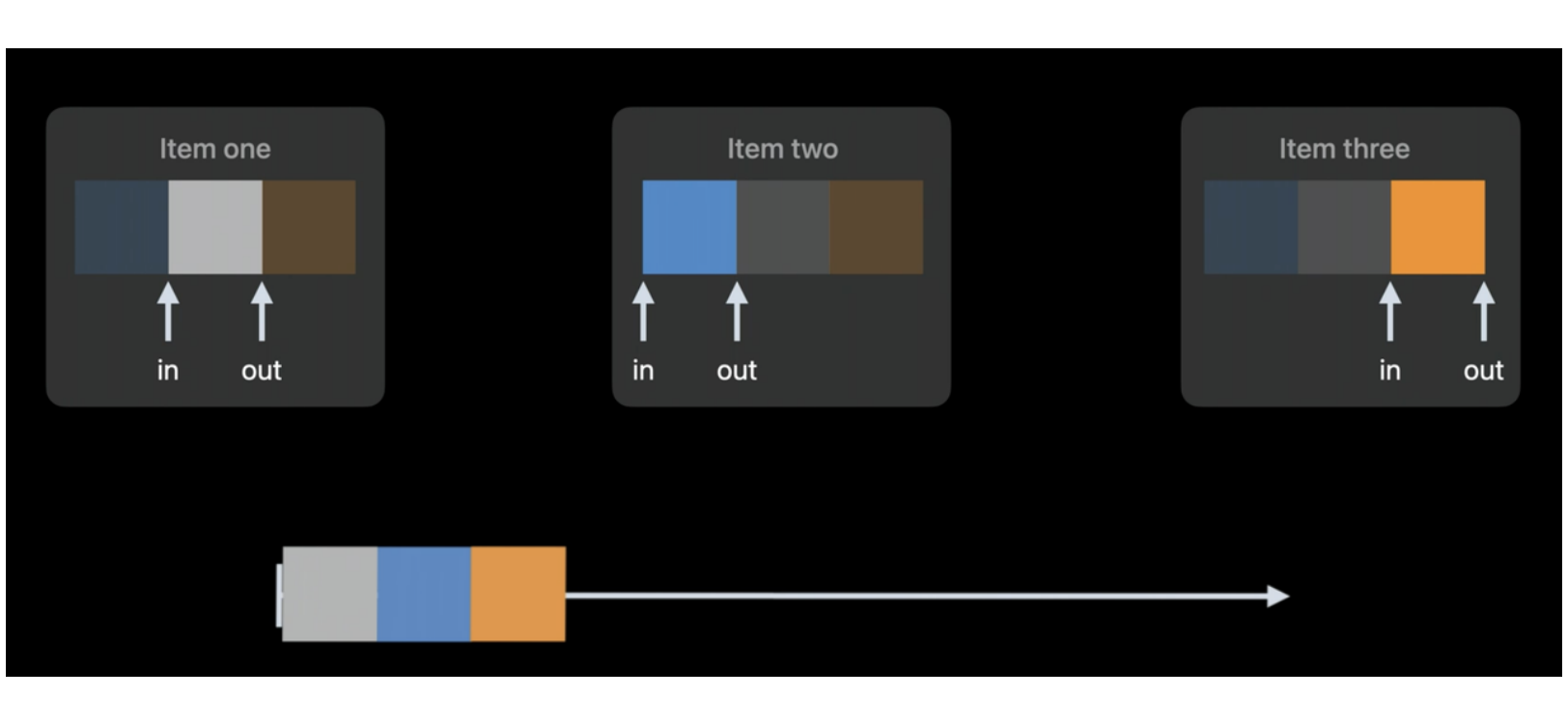
资源切割技巧

此外，Simon 还分享了一个资源切割的技巧。简单地说，就是可以在同一资源上划分多个片段进行播放。在代码中就是在一个 AVAsset 中实例化出多个 AVPlayerItem 播放。

这么做的好处是可以动态地选择需要播放的片段和调整播放顺序，相当于可以使用代码重新剪辑资源。

实现这个功能需要依赖 AVPlayerItem 的 `seekToTime` 和 `forwardPlaybackEndTime` 这两个方法。使用 `seekToTime` 设置起始点，`forwardPlaybackEndTime` 设置终止点。再通过 AVQueuePlayer 将分割的片段连起来播放。

比如在下图中，我们用三个色块（蓝、灰、橙）代表一个资源的三个片段，正常的播放顺序为蓝灰橙。而如果我们使用上述方法将资源重新排列，就能把播放顺序排为灰蓝橙。



代码示例：

```
let asset = AVAsset(url: url)

let startTimes: [CMTime] = [startTime1, startTime2, startTime3]
let endTimes: [CMTime] = [endTime1, endTime2, endTime3]

let item0 = AVPlayerItem(asset: asset)
item0.seek(to: startTimes[0], completionHandler: nil)
item0.forwardPlaybackEndTime = endTimes[0]

let item1 = AVPlayerItem(asset: asset)
item1.seek(to: startTimes[1], completionHandler: nil)
item1.forwardPlaybackEndTime = endTimes[1]

let item2 = AVPlayerItem(asset: asset)
item2.seek(to: startTimes[2], completionHandler: nil)
item2.forwardPlaybackEndTime = endTimes[2]

let player = AVQueuePlayer()

// 此处改变了原本的播放顺序
player.insert(item1, after: nil)
player.insert(item2, after: item1)
player.insert(item0, after: item2)

player.play()
```

结语

使用无缝切换技术可以让我们为用户提供一个流畅的音视频体验。无缝切换技术本质上还是减少或避免因参数变化而导致的切换性能损耗。它可能不是众多新功能中改动最大的，但有一点可以肯定的是，无缝切换是最能体现苹果极致的工匠精神的特性之一。

关注我们

我们是「老司机技术周报」，一个持续追求精品 iOS 内容的技术公众号。欢迎关注。

老司机技术周报

微信扫描二维码，关注我的公众号

关注有礼，关注【老司机技术周报】，回复「WWDC」，领取 《WWDC20 内参》

这个作品真棒，我要支持一下！

赞赏

赞 收藏 分享 转载

WWDC21 内参
一年一度的 WWDC 又来啦！今年官方一共放出了 212 个 Session 内容，我们筛选了其中 168 个 S...
已经订阅

1条评论

老司机技术周报 #1
对文中有疑问的可以加我们的微信 iTDriverr，备注「WWDC21」拉你进入读者群交流哦。
7 月前 赞

写下你的评论