

# Moving to AV Kit and AV Foundation

Session 606

**Sam Bushell**

Media Frameworks Architect

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

# Introduction

- A word on QuickTime and QTKit
- New AV Kit API
- Travel Guide to AV Foundation and AV Kit

# AV Foundation

- We've been building a new media infrastructure
- Common on iOS and OS X
- Focused on modern media formats
- Benefits from deep media experience building QuickTime

# Mac OS 7, 8, 9 and Mac OS X

1991

QuickTime

# Mac OS X 10.4 Tiger

QTKit introduced with QuickTime 7.0



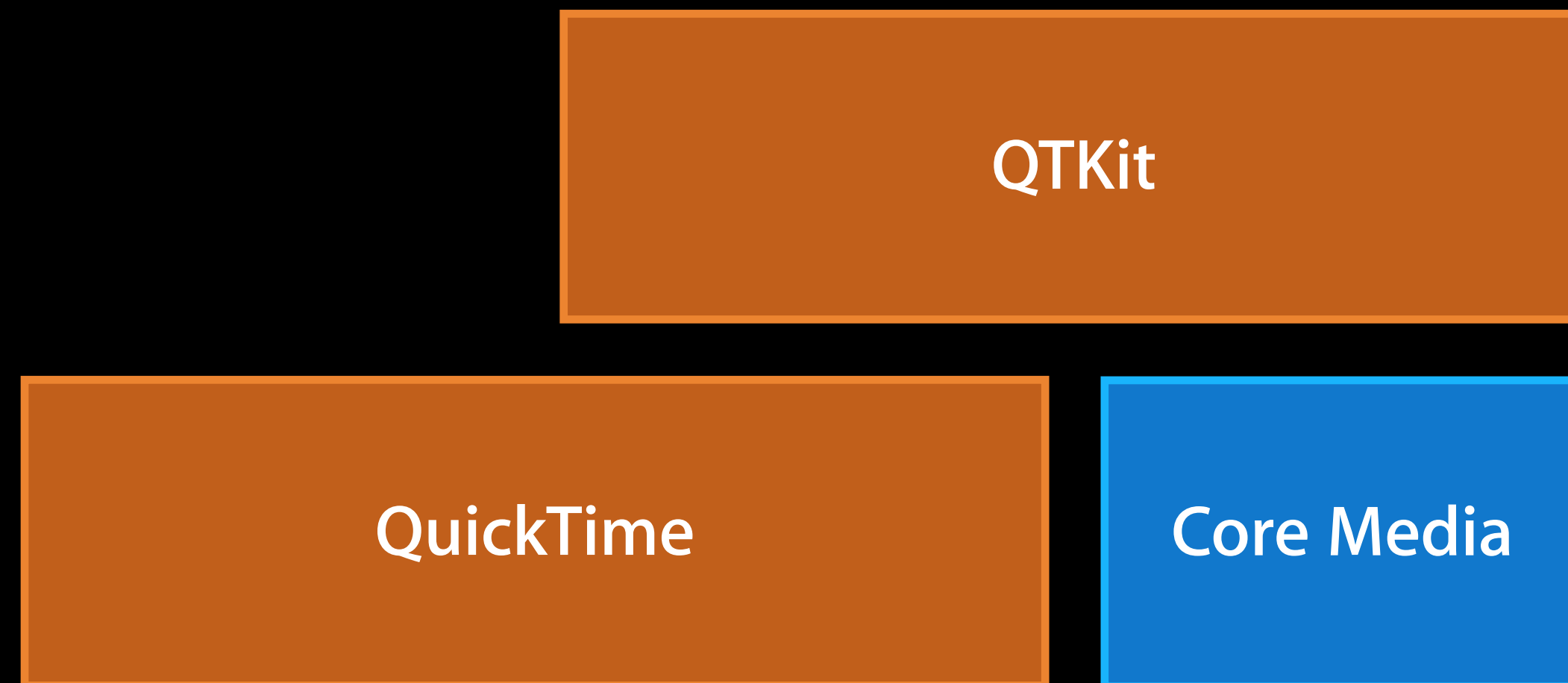
QTKit

The diagram consists of two orange rectangular boxes stacked vertically. The top box is labeled 'QTKit' and the bottom box is labeled 'QuickTime'. Both boxes have a thin orange border.

QuickTime

# Mac OS X 10.6 Snow Leopard

Optimized H.264 + AAC: QTMovieOpenForPlaybackAttribute



# Mac OS X 10.7 Lion

AV Foundation introduced

QTKit

AV Foundation

QuickTime

Core Media

# OS X 10.8 Mountain Lion

Video Toolbox API introduced

QTKit

AV Foundation

QuickTime

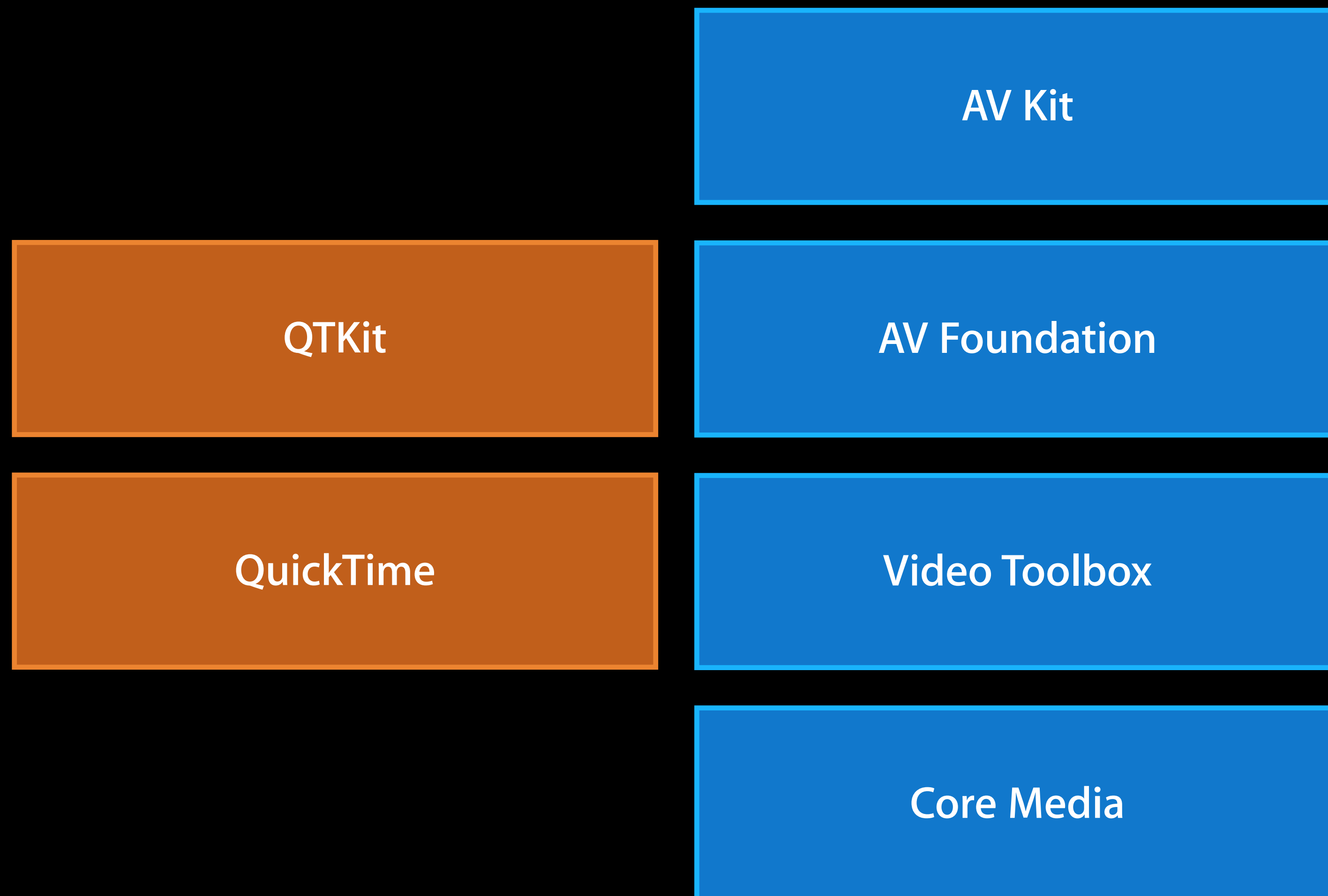
Video Toolbox

Core Media



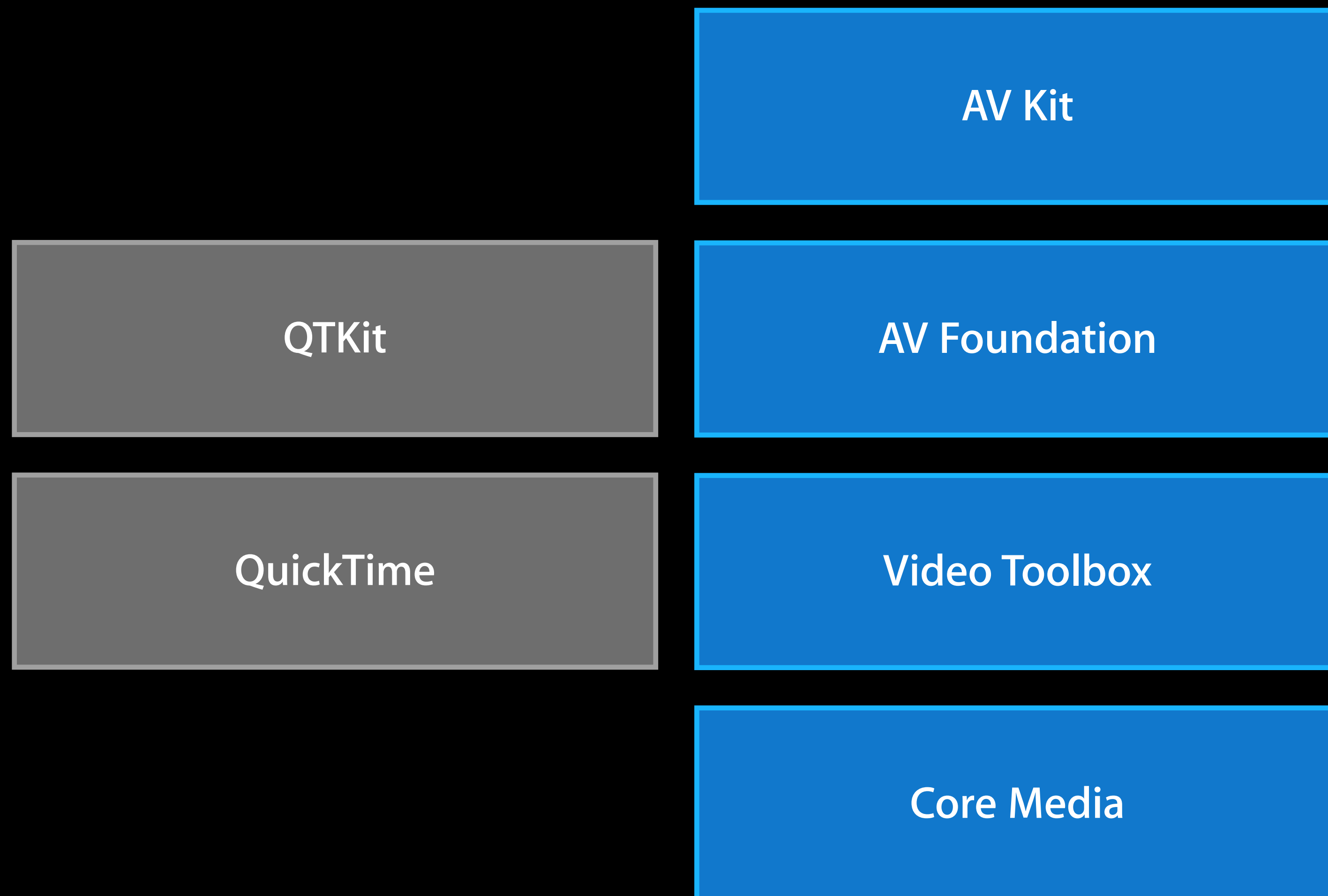
# OS X 10.9 Mavericks

AV Kit introduced



# OS X 10.9 Mavericks

AV Kit introduced



# QuickTime and QTKit APIs Are Deprecated

- QuickTime.framework and QTKit.framework APIs deprecated in OS X 10.9
- APIs are marked as deprecated in header files
- Code will still compile, but with deprecation warnings
- Your apps will still run

```
27  
28 SetIdentityMatrix(&matrix);           ⚠ 'SetIdentityMatrix' is deprecated: first deprecated in OS X 10.9  
29 SetMovieMatrix(dstMovie, &matrix);    ⚠ 'SetMovieMatrix' is deprecated: first deprecated in OS X 10.9  
30 SetMovieClipRgn(dstMovie, NULL);      ⚠ 'SetMovieClipRgn' is deprecated: first deprecated in OS X 10.9  
31  
32 result = BeginMediaEdits(dstMedia);    ⚠ 'BeginMediaEdits' is deprecated: first deprecated in OS X 10.9  
33
```



# QuickTime and QTKit APIs Are Deprecated

- QuickTime.framework and QTKit.framework APIs deprecated in OSX 10.9
- APIs are marked as deprecated in header files
- Code will still compile, but with deprecation warnings
- Your apps will still run

```
27
28 SetIdentityMatrix(&matrix);      ⚠ 'SetIdentityMatrix' is deprecated: first deprecated in OS X 10.9
29 SetMovieMatrix(dstMovie, &matrix); ⚠ 'SetMovieMatrix' is deprecated: first deprecated in OS X 10.9
30 SetMovieClipRgn(dstMovie, NULL); ⚠ 'SetMovieClipRgn' is deprecated: first deprecated in OS X 10.9
31
32 result = BeginMediaEdits(dstMedia); ⚠ 'BeginMediaEdits' is deprecated: first deprecated in OS X 10.9
33
```

# QuickTime Movie Format

Still supported

- AV Foundation and QuickTime Player still use the QuickTime Movie file format
- Apple is deprecating the QuickTime 7 APIs, **not the file format**



**.MOV**

# QuickTime Movie Format

Still supported

- AV Foundation and QuickTime Player still use the QuickTime Movie file format
- Apple is deprecating the QuickTime 7 APIs, **not the file format**



.MOV



# QuickTime Movie Format

Still supported

- AV Foundation and QuickTime Player still use the QuickTime Movie file format
- Apple is deprecating the QuickTime 7 APIs, **not the file format**



.MOV



.MP4







# *AV* Foundation

**AV Foundation**

**QuickTime**

AV Foundation	QuickTime
Modern foundation	Carbon, QuickDraw, Handles

AV Foundation	QuickTime
Modern foundation	Carbon, QuickDraw, Handles
APIs designed for clients	APIs expose implementation

AV Foundation	QuickTime
Modern foundation	Carbon, QuickDraw, Handles
APIs designed for clients	APIs expose implementation
Factored	Monolithic

AV Foundation	QuickTime
Modern foundation	Carbon, QuickDraw, Handles
APIs designed for clients	APIs expose implementation
Factored	Monolithic
Multithreaded	Mostly main-thread-only

AV Foundation	QuickTime
Modern foundation	Carbon, QuickDraw, Handles
APIs designed for clients	APIs expose implementation
Factored	Monolithic
Multithreaded	Mostly main-thread-only
Hardware accelerated	—

AV Foundation	QuickTime
Modern foundation	Carbon, QuickDraw, Handles
APIs designed for clients	APIs expose implementation
Factored	Monolithic
Multithreaded	Mostly main-thread-only
Hardware accelerated	—
Power efficient	—



AV Foundation	QuickTime
Modern foundation	Carbon, QuickDraw, Handles
APIs designed for clients	APIs expose implementation
Factored	Monolithic
Multithreaded	Mostly main-thread-only
Hardware accelerated	—
Power efficient	—
64-bit native	32-bit only

# Supported Media Types



- Video
- Audio
- Closed captions and subtitles
- Chapters
- Timecode

# Not Supported by AV Foundation

Some examples



QuickTime VR

RTP Streaming

QT Effects and Filters

Sprite Tracks and Wired Sprites

Flash Tracks

Music (MIDI) Tracks

SMIL

# Supported Codecs



- Delivery codecs
  - H.264, AAC, JPEG
- Mezzanine codecs
  - Apple ProRes, LPCM
- Camera device codecs
  - MPEG-1, MPEG-2, MPEG-4, H.263, DV...

# Not Supported by AV Foundation



Cinepak ("Compact Video")  
Animation ("RLE")  
Video ("Road Pizza")  
Graphics ("SMC")  
Sorenson Video  
Sorenson Video 3  
Motion JPEG A  
Motion JPEG B  
H.261  
Windows RAW  
Microsoft Video 1  
Pixlet  
MACE 3:1  
MACE 6:1  
QDesign Audio  
QDesign Audio 2  
1-bit Indexed-Color RGB  
2-bit Indexed-Color RGB

4-bit Indexed-Color RGB  
8-bit Indexed-Color RGB  
16-bit Direct-Color RGB  
1-bit Grayscale  
2-bit Grayscale  
4-bit Grayscale  
SGI  
MacPaint  
BMP  
FLC  
FlashPix  
JPEG 2000  
PDF  
Photo CD  
PNG  
TGA  
TIFF  
Blit Codec

Curve Rasterizer  
Quickdraw Codec  
Blend Effect  
Blur Filter  
Brightness and Contrast  
Channel Compositor  
Chroma Key Effect  
Cloud Generator  
Cross Fade Effect  
Edge Detection Filter  
Emboss Filter  
Fire Generator  
Film Noise Filter  
Alpha Gain Filter  
General Convolution  
Glass Distortion Filter  
HSL Balance Filter  
Lens Flare Filter

Gradient Wipe Effect  
Implode Effect  
Push Effect  
RGB Balance Filter  
Ripple Filter  
Sharpen Filter  
Slide Effect  
SMPTE Iris Effect  
SMTPE Radial Effect  
SMTPE Matrix Wipe Effect  
Wipe Effect  
Color Style Filter  
ColorSync Filter  
Travelling Matte Effect  
Explode Effect  
Zoom Effect



Your Baby's First Steps.mov

Sorenson Video 3

QDesign Audio 2



Your Baby's First Steps.mov

Sorenson Video 3

QDesign Audio 2



Your Baby's First Steps.mov

Sorenson Video 3

QDesign Audio 2





Your Baby's First Steps.mov

Sorenson Video 3

QDesign Audio 2

**H.264**

**AAC**

# QTMovieModernizer



- Automatically run by QuickTime Player upon discovery of legacy codecs
  - Works with third-party QuickTime codec components
- New API in OS X 10.9 so that you can do the same in your apps
- Produces a new copy in an AV Foundation-supported format:
  - H.264 + AAC
  - Apple ProRes 422 + Linear PCM
  - Apple ProRes 4444 + Linear PCM
- Delivered as part of QTKit

QTKit

QuickTime

AV Kit

QTKit

AV Foundation

QuickTime

Video Toolbox

Core Media

# Introducing AV Kit

**Stefan Hafeneger**  
Media Systems Engineer

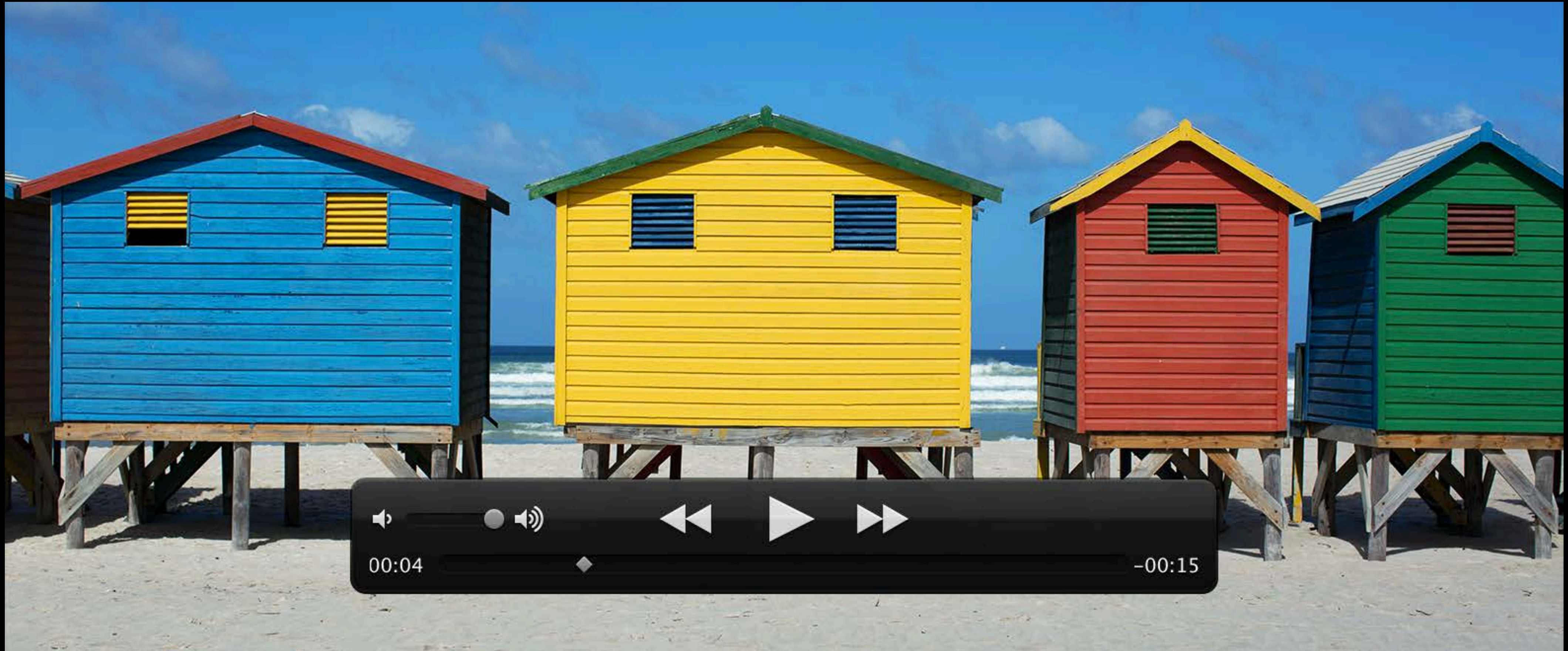
# AV Kit



AVKit.framework



# AVPlayerView





*Demo*

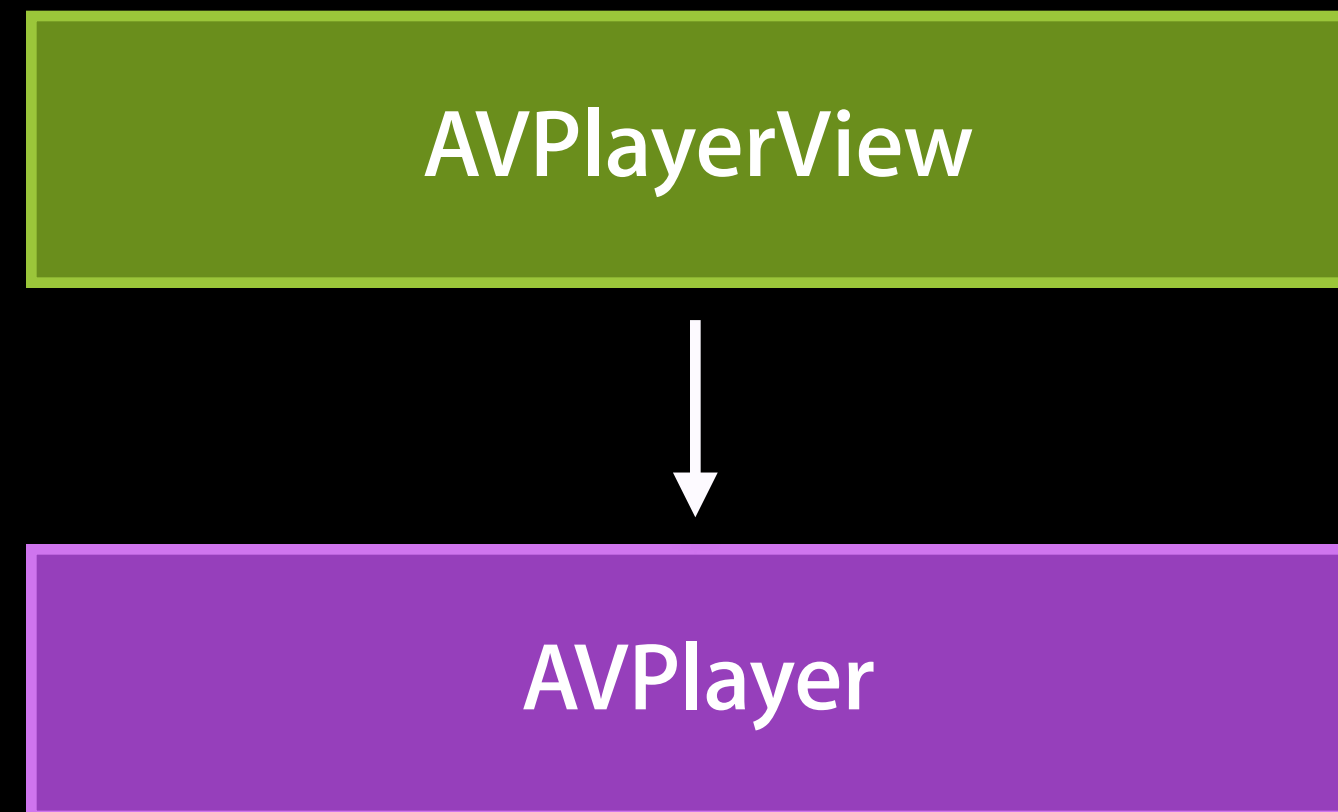
AVPlayerView



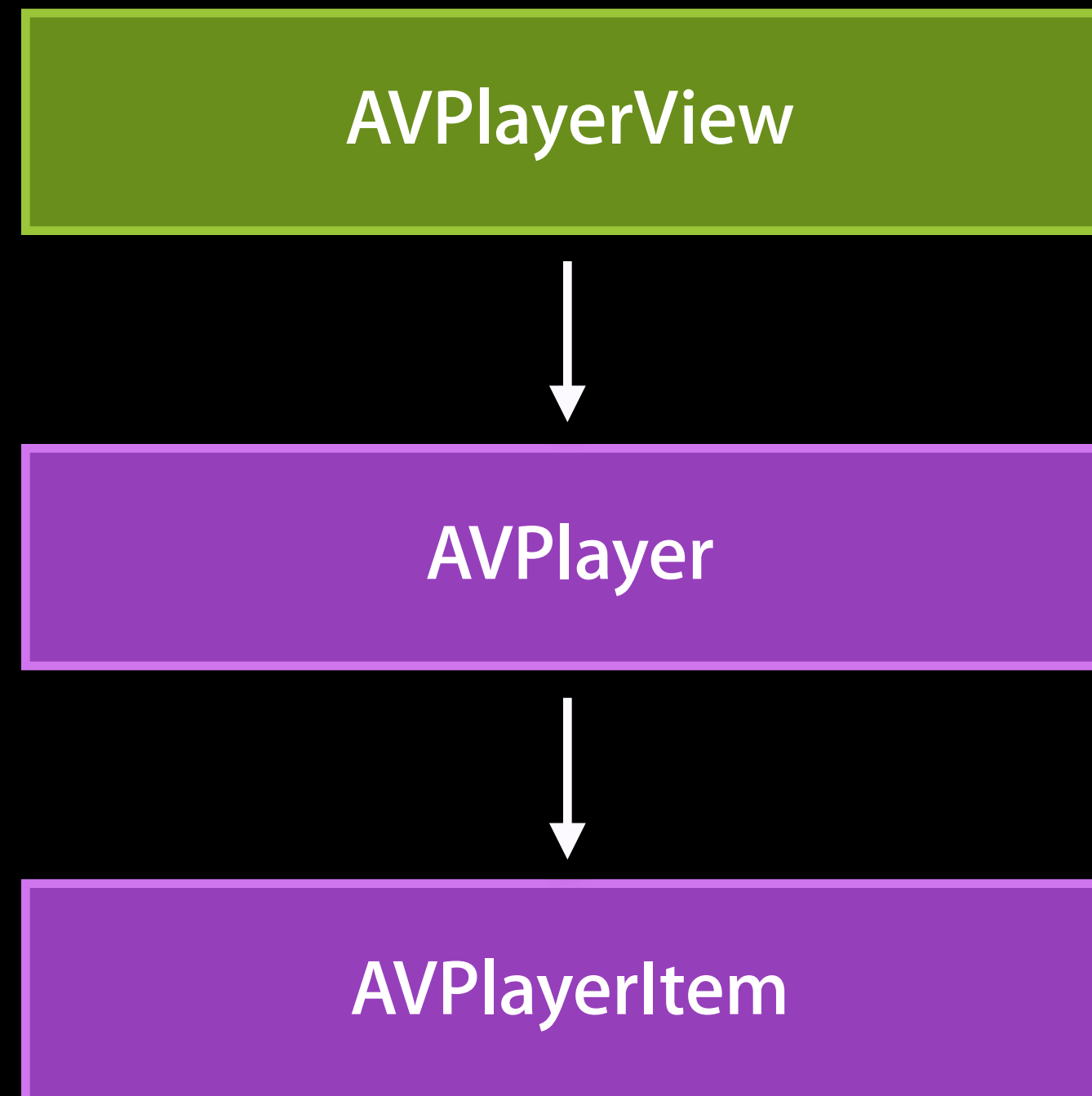
# AVPlayerView and AV Foundation

AVPlayerView

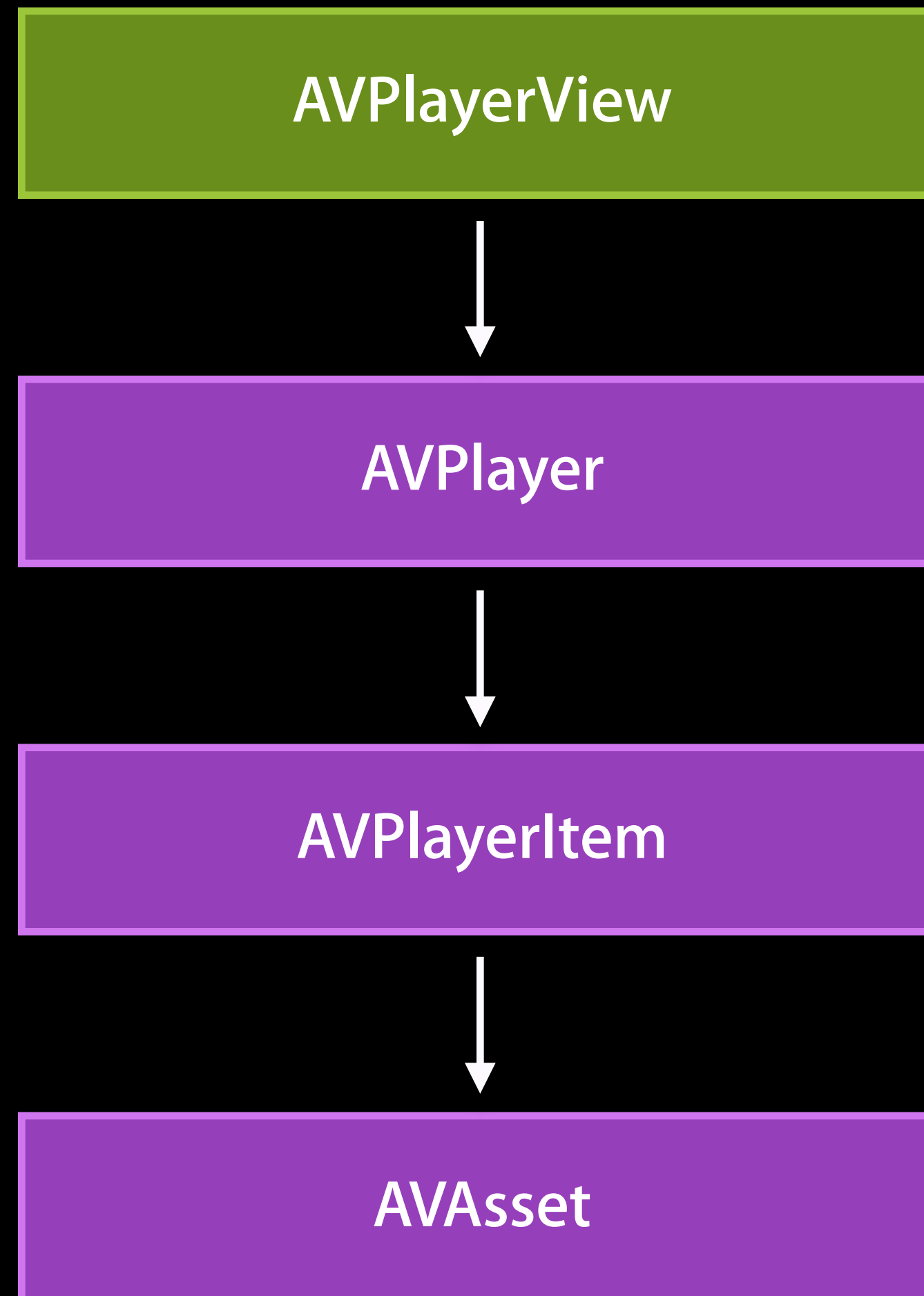
# AVPlayerView and AV Foundation



# AVPlayerView and AV Foundation



# AVPlayerView and AV Foundation



# Providing Content

## Steps to provide content for AVPlayerView

// 1. Create asset from URL.

```
AVAsset *asset = [AVAsset assetWithURL:URL];
```

// 2. Create player item for asset.

```
AVPlayerItem *playerItem = [AVPlayerItem playerItemWithAsset:asset];
```

// 3. Create player with player item.

```
AVPlayer *player = [AVPlayer playerWithPlayerItem:playerItem];
```

// 4. Associate player with player view.

```
[playerView setPlayer:player];
```

# Providing Content

## Steps to provide content for AVPlayerView

```
// 1. Create asset from URL.
```

```
AVAsset *asset = [AVAsset assetWithURL:URL];
```

```
// 2. Create player item for asset.
```

```
AVPlayerItem *playerItem = [AVPlayerItem playerItemWithAsset:asset];
```

```
// 3. Create player with player item.
```

```
AVPlayer *player = [AVPlayer playerWithPlayerItem:playerItem];
```

```
// 4. Associate player with player view.
```

```
[playerView setPlayer:player];
```

# Providing Content

## Steps to provide content for AVPlayerView

// 1. Create asset from URL.

```
AVAsset *asset = [AVAsset assetWithURL:URL];
```

// 2. Create player item for asset.

```
AVPlayerItem *playerItem = [AVPlayerItem playerItemWithAsset:asset];
```

// 3. Create player with player item.

```
AVPlayer *player = [AVPlayer playerWithPlayerItem:playerItem];
```

// 4. Associate player with player view.

```
[playerView setPlayer:player];
```

# Providing Content

## Steps to provide content for AVPlayerView

// 1. Create asset from URL.

```
AVAsset *asset = [AVAsset assetWithURL:URL];
```

// 2. Create player item for asset.

```
AVPlayerItem *playerItem = [AVPlayerItem playerItemWithAsset:asset];
```

// 3. Create player with player item.

```
AVPlayer *player = [AVPlayer playerWithPlayerItem:playerItem];
```

// 4. Associate player with player view.

```
[playerView setPlayer:player];
```



# Providing Content

## Steps to provide content for AVPlayerView

// 1. Create asset from URL.

```
AVAsset *asset = [AVAsset assetWithURL:URL];
```

// 2. Create player item for asset.

```
AVPlayerItem *playerItem = [AVPlayerItem playerItemWithAsset:asset];
```

// 3. Create player with player item.

```
AVPlayer *player = [AVPlayer playerWithPlayerItem:playerItem];
```

// 4. Associate player with player view.

```
[playerView setPlayer:player];
```

# Providing Content

## Steps to provide content for AVPlayerView

// 1. Create asset from URL.

```
AVAsset *asset = [AVAsset assetWithURL:URL];
```

// 2. Create player item for asset.

```
AVPlayerItem *playerItem = [AVPlayerItem playerItemWithAsset:asset];
```

// 3. Create player with player item.

```
AVPlayer *player = [AVPlayer playerWithPlayerItem:playerItem];
```

// 4. Associate player with player view.

```
[playerView setPlayer:player];
```

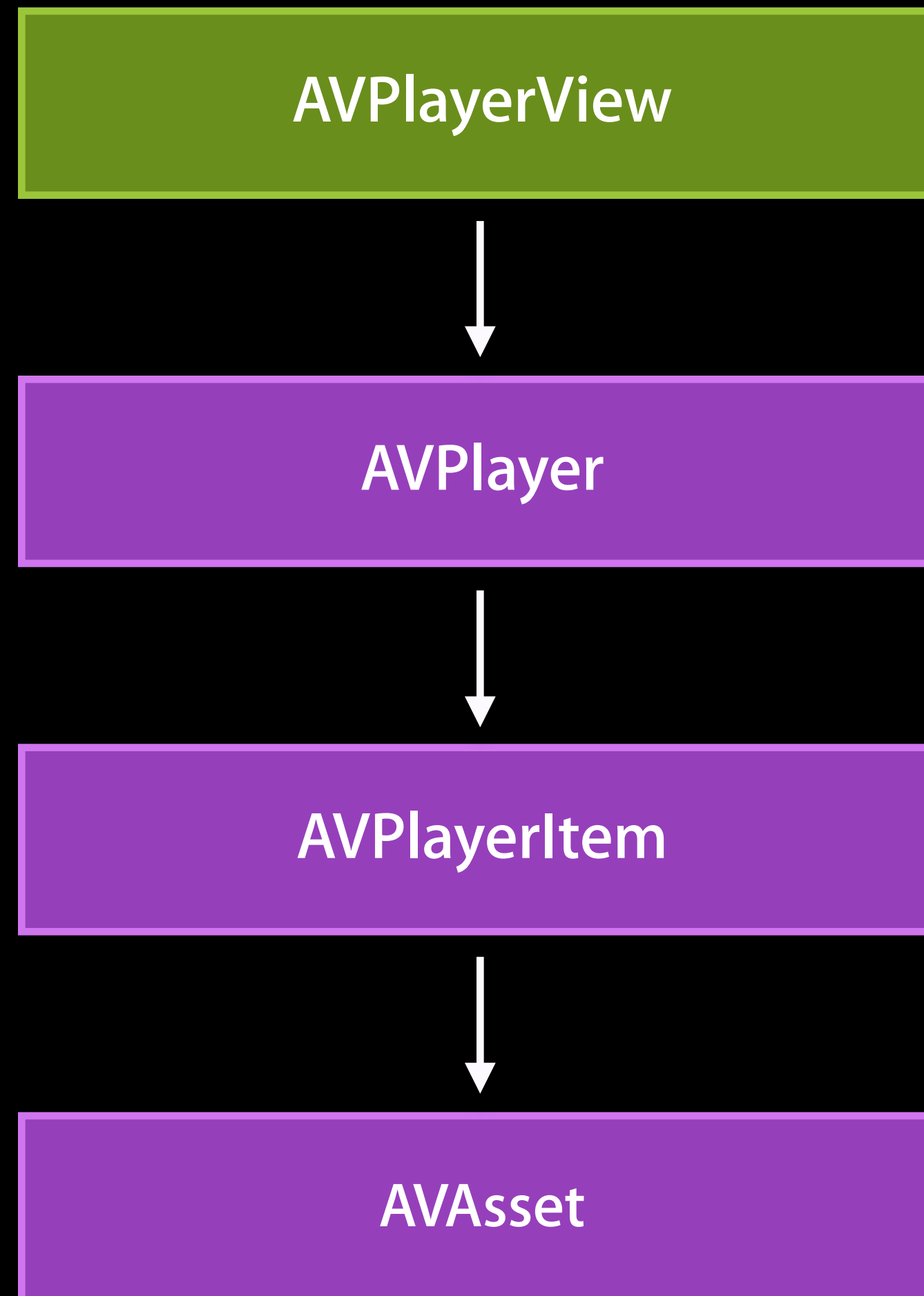
# Providing Content

One step to provide content for *AVPlayerView*

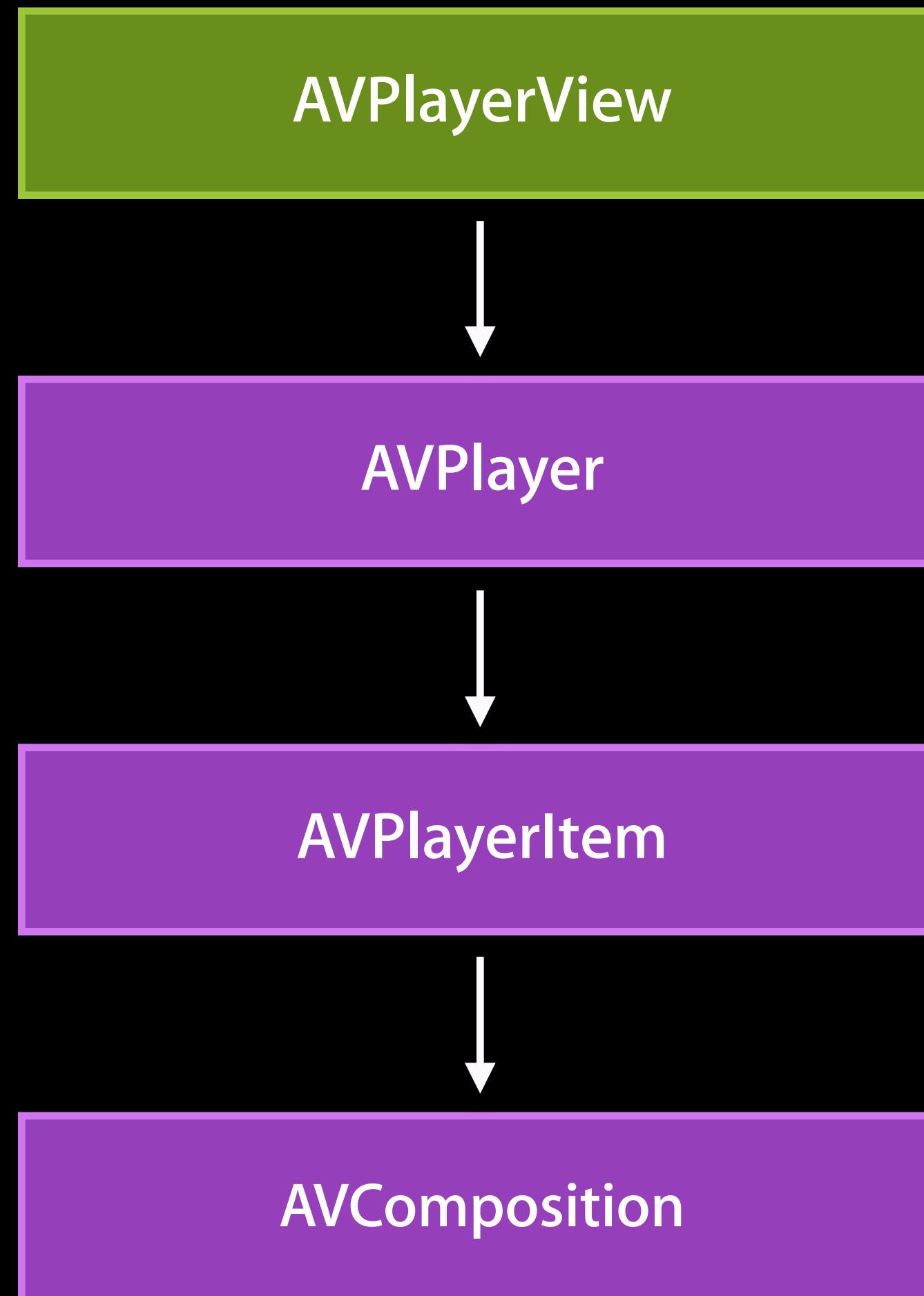
// All four steps in one line of code.

```
[playerView setPlayer:[AVPlayer playerWithURL:URL]];
```

# AVPlayerView and AV Foundation



# AVPlayerView and AV Foundation





# Defining the Look

## AVPlayerView's controlsStyle property





# Defining the Look

AVPlayerView's controlsStyle property





# Defining the Look

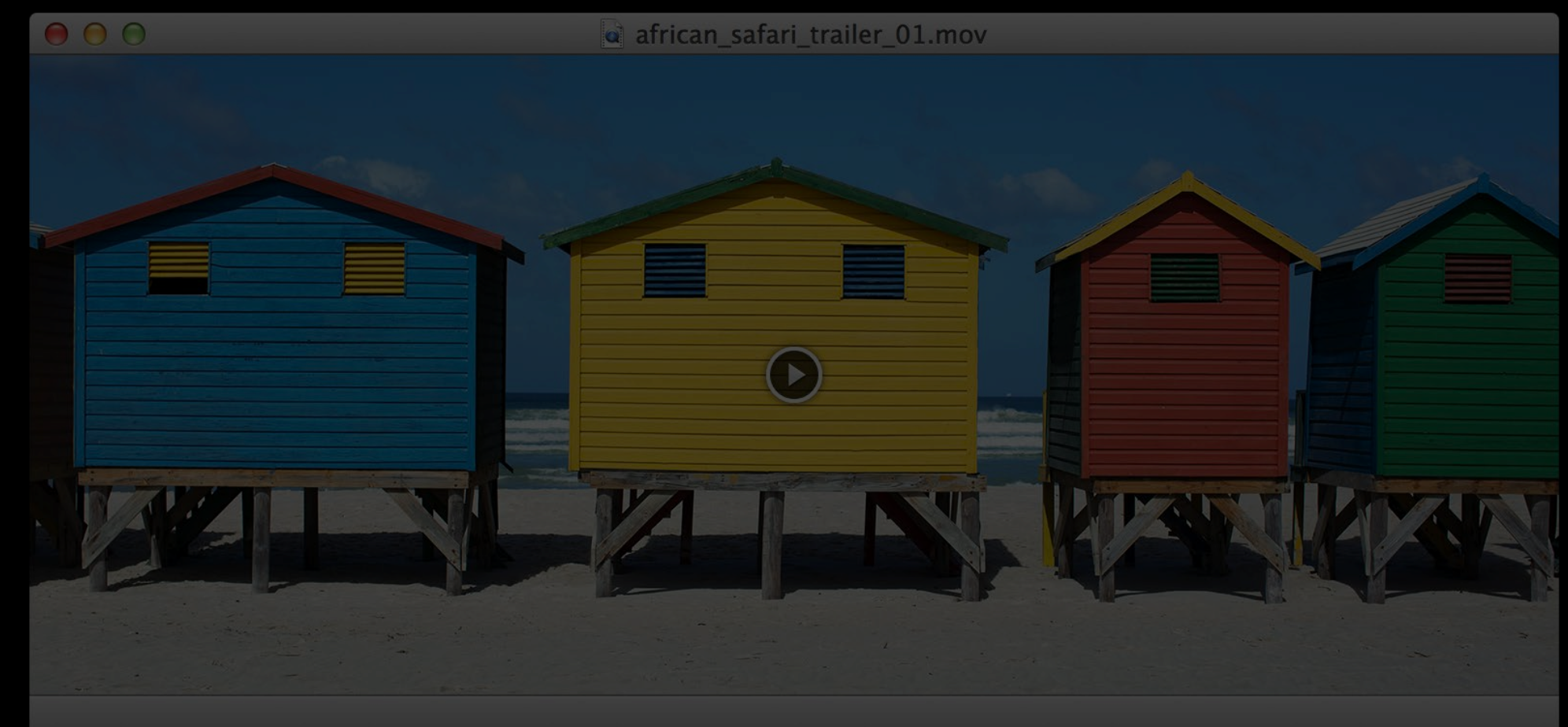
AVPlayerView's controlsStyle property





# Defining the Look

AVPlayerView's controlsStyle property





# Defining the Look

AVPlayerView's controlsStyle property





# Defining the Look

## AVPlayerView's controlsStyle property





# Defining the Look

## AVPlayerView's controlsStyle property





# Defining the Look

AVPlayerView's controlsStyle property





# Defining the Look

## AVPlayerView's controlsStyle property





# Defining the Look

AVPlayerView's controlsStyle property





# Defining the Look

AVPlayerView's controlsStyle property





# Defining the Look

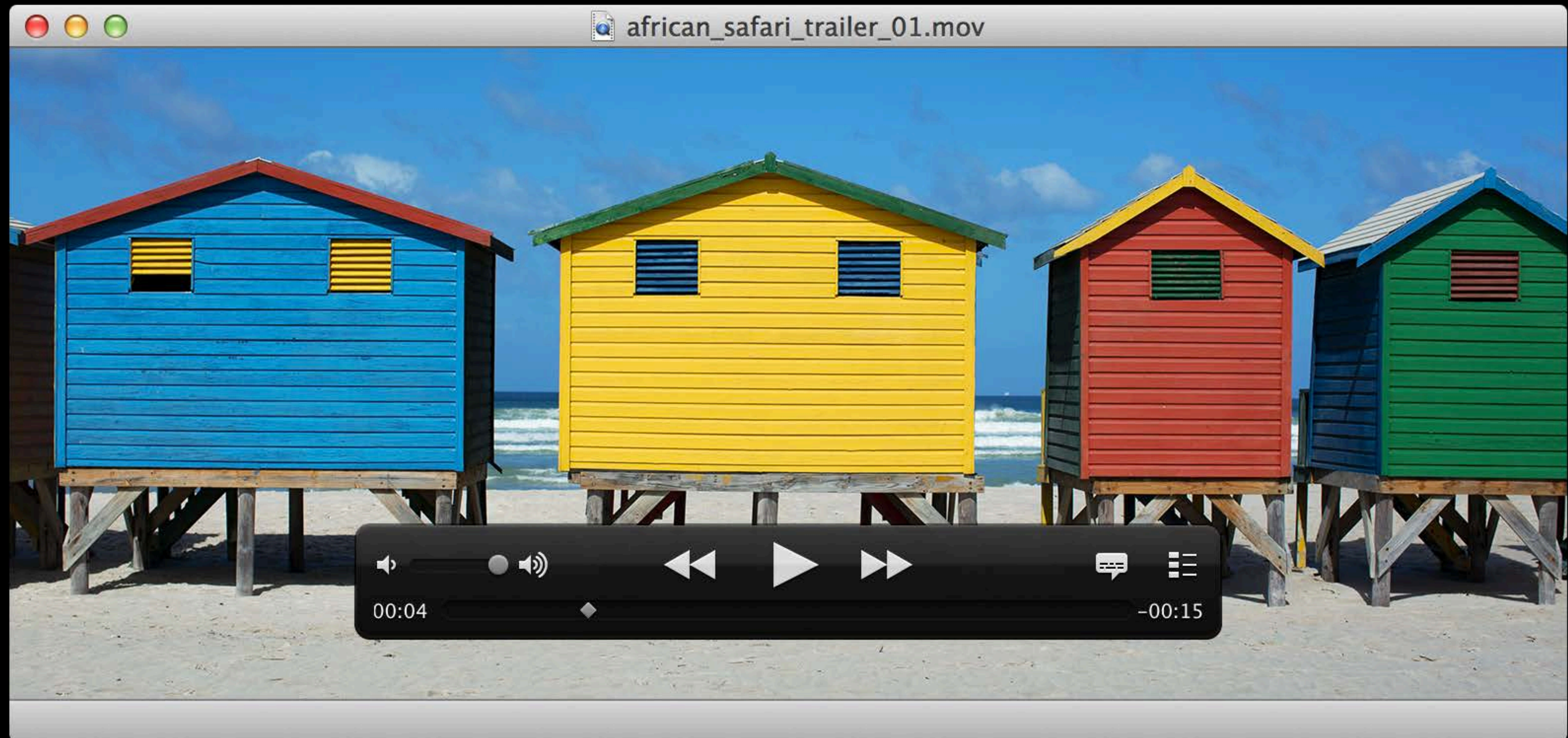
AVPlayerView's controlsStyle property





# Dynamic Controls

Chapters, languages and subtitles





# Dynamic Controls

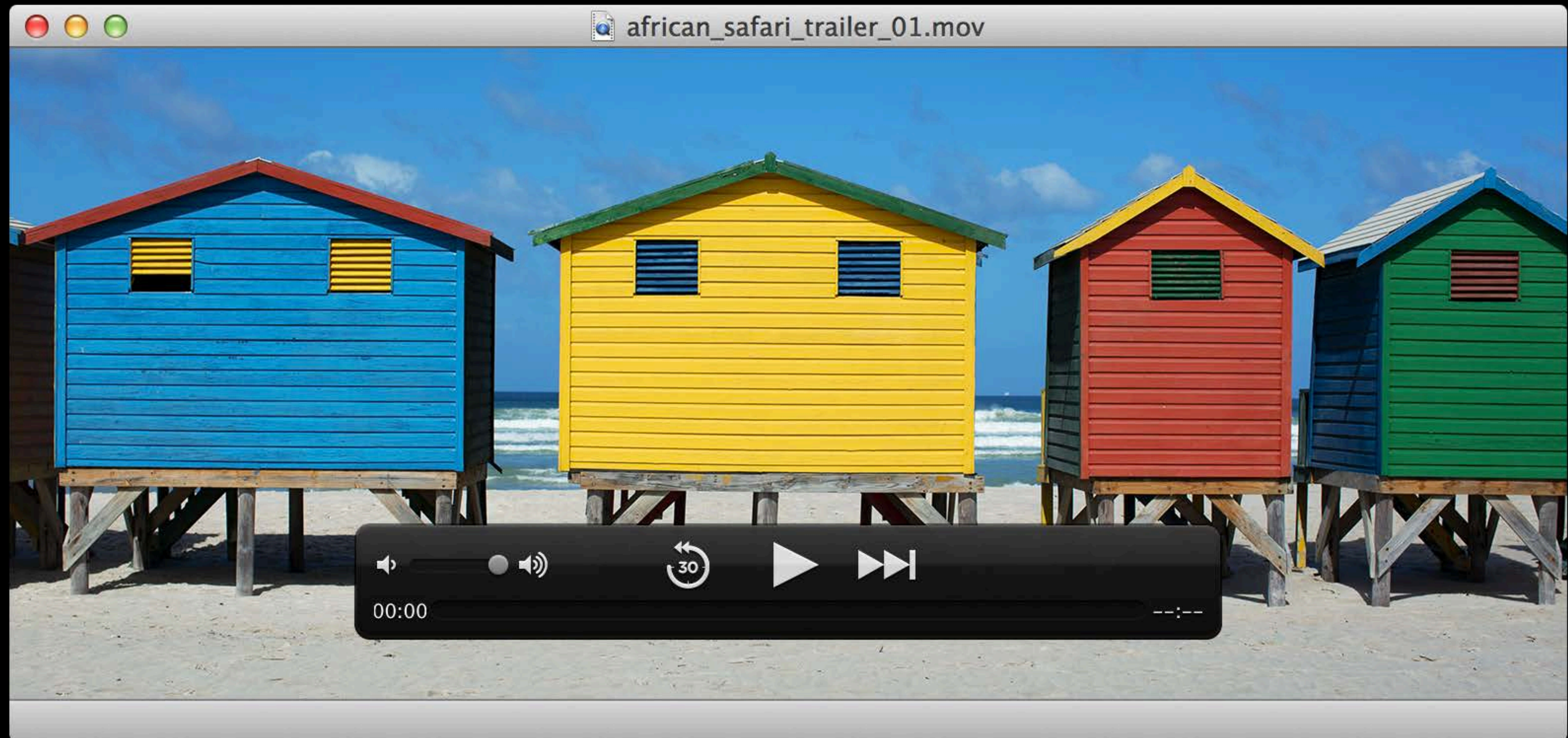
## Chapters, languages and subtitles





# Dynamic Controls

## Streaming media





# Dynamic Controls

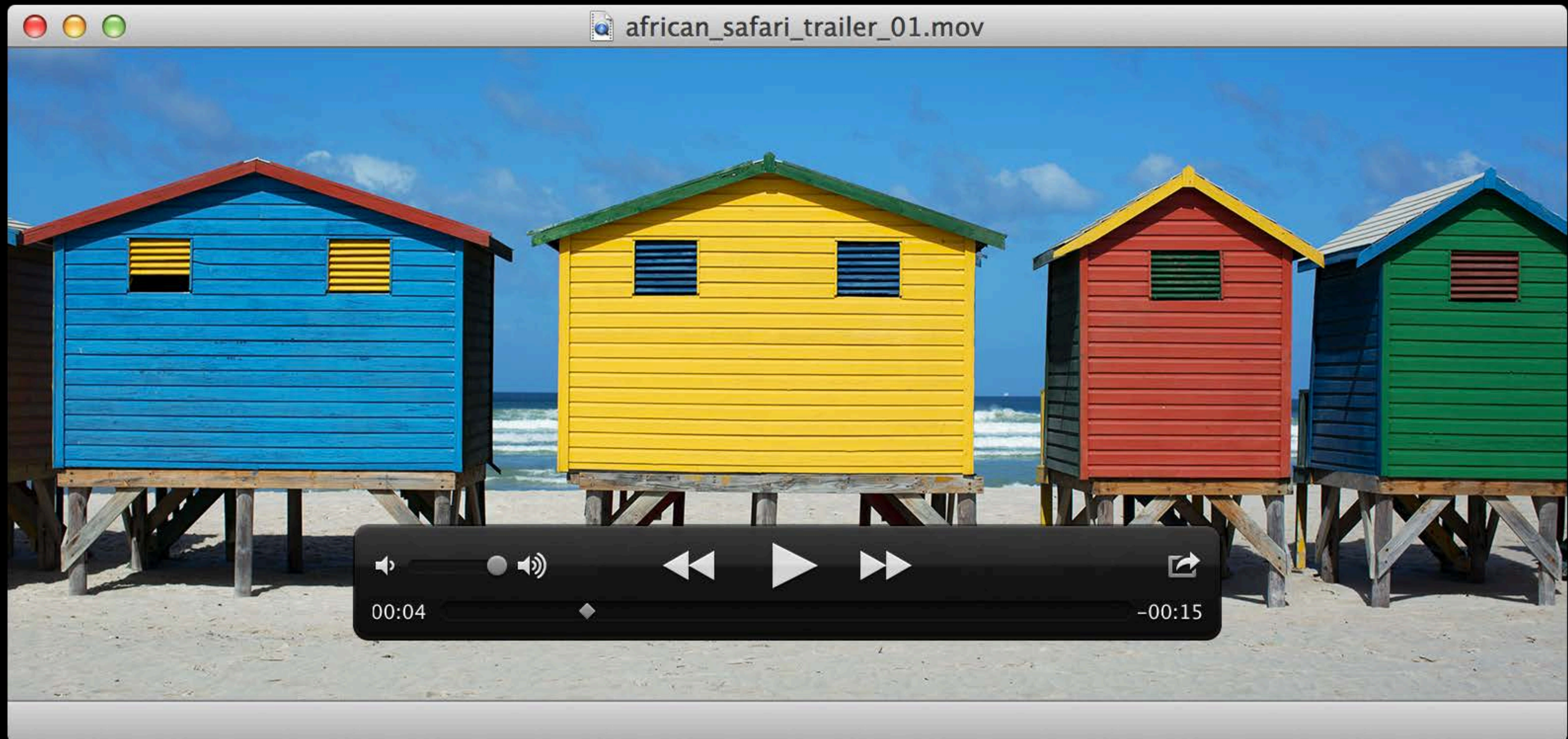
## Streaming media





# Customization

## Sharing service button





# Customization

## Sharing service button





# Trimming

## Show trim user interface

```
// Check whether current item can be trimmed.
if ([playerView canBeginTrimming])
{
    // Show trim user interface.
    [playerView beginTrimmingWithCompletionHandler:NULL];
}
```

# Trimming

## Show trim user interface

```
// Check whether current item can be trimmed.
if ([playerView canBeginTrimming])
{
    // Show trim user interface.
    [playerView beginTrimmingWithCompletionHandler:NULL];
}
```

# Trimming

## Show trim user interface

```
// Check whether current item can be trimmed.
if ([playerView canBeginTrimming])
{
    // Show trim user interface.
    [playerView beginTrimmingWithCompletionHandler:NULL];
}
```

# Trimming

Handle trim result after trim user interface is dismissed

```
// Check whether current item can be trimmed.
if ([playerView canBeginTrimming])
{
    // Show trim user interface.
    [playerView beginTrimmingWithCompletionHandler:^(AVPlayerViewTrimResult
result) {
        // Handle trim result.
        if (result == AVPlayerViewTrimOKButton)
        {
            CMTime inPoint = [playerItem reversePlaybackEndTime];
            CMTime outPoint = [playerItem forwardPlaybackEndTime];
        }
        else if (result == AVPlayerViewTrimCancelButton)
        { ... }
    }];
}
```

# Trimming

Handle trim result after trim user interface is dismissed

```
// Check whether current item can be trimmed.
if ([playerView canBeginTrimming])
{
    // Show trim user interface.
    [playerView beginTrimmingWithCompletionHandler:^(AVPlayerViewTrimResult
result) {
        // Handle trim result.
        if (result == AVPlayerViewTrimOKButton)
        {
            CMTime inPoint = [playerItem reversePlaybackEndTime];
            CMTime outPoint = [playerItem forwardPlaybackEndTime];
        }
        else if (result == AVPlayerViewTrimCancelButton)
        { ... }
    }];
}
```

# Trimming

Handle trim result after trim user interface is dismissed

```
// Check whether current item can be trimmed.
if ([playerView canBeginTrimming])
{
    // Show trim user interface.
    [playerView beginTrimmingWithCompletionHandler:^(AVPlayerViewTrimResult
result) {
        // Handle trim result.
        if (result == AVPlayerViewTrimOKButton)
        {
            CMTime inPoint = [playerItem reversePlaybackEndTime];
            CMTime outPoint = [playerItem forwardPlaybackEndTime];
        }
        else if (result == AVPlayerViewTrimCancelButton)
        { ... }
    }];
}
```

# Trimming

## Handle trim result after trim user interface is dismissed

```
// Check whether current item can be trimmed.
if ([playerView canBeginTrimming])
{
    // Show trim user interface.
    [playerView beginTrimmingWithCompletionHandler:^(AVPlayerViewTrimResult
result) {
        // Handle trim result.
        if (result == AVPlayerViewTrimOKButton)
        {
            CMTime inPoint = [playerItem reversePlaybackEndTime];
            CMTime outPoint = [playerItem forwardPlaybackEndTime];
        }
        else if (result == AVPlayerViewTrimCancelButton)
        { ... }
    }];
}
```

# Trimming

## Handle trim result after trim user interface is dismissed

```
// Check whether current item can be trimmed.
if ([playerView canBeginTrimming])
{
    // Show trim user interface.
    [playerView beginTrimmingWithCompletionHandler:^(AVPlayerViewTrimResult
result) {
        // Handle trim result.
        if (result == AVPlayerViewTrimOKButton)
        {
            CMTime inPoint = [playerItem reversePlaybackEndTime];
            CMTime outPoint = [playerItem forwardPlaybackEndTime];
        }
        else if (result == AVPlayerViewTrimCancelButton)
        { ... }
    }];
}
```



# Trimming

## Export trim selection

```
// Get trim in and out points.
```

```
CMTime inPoint = [playerItem reversePlaybackEndTime];
```

```
CMTime outPoint = [playerItem forwardPlaybackEndTime];
```

```
// Set time range on asset export session.
```

```
CMTimeRange timeRange = CMTimeRangeFromTimeToTime(inPoint, outPoint);
```

```
[assetExportSession setTimeRange:timeRange];
```

# Trimming

## Export trim selection

```
// Get trim in and out points.
```

```
CMTime inPoint = [playerItem reversePlaybackEndTime];
```

```
CMTime outPoint = [playerItem forwardPlaybackEndTime];
```

```
// Set time range on asset export session.
```

```
CMTimeRange timeRange = CMTimeRangeFromTimeToTime(inPoint, outPoint);
```

```
[assetExportSession setTimeRange:timeRange];
```

# Trimming

## Export trim selection

```
// Get trim in and out points.
```

```
CMTime inPoint = [playerItem reversePlaybackEndTime];
```

```
CMTime outPoint = [playerItem forwardPlaybackEndTime];
```

```
// Set time range on asset export session.
```

```
CMTimeRange timeRange = CMTimeRangeFromTimeToTime(inPoint, outPoint);
```

```
[assetExportSession setTimeRange:timeRange];
```

# Trimming

## Export trim selection

```
// Get trim in and out points.
```

```
CMTime inPoint = [playerItem reversePlaybackEndTime];
```

```
CMTime outPoint = [playerItem forwardPlaybackEndTime];
```

```
// Set time range on asset export session.
```

```
CMTimeRange timeRange = CMTimeRangeFromTimeToTime(inPoint, outPoint);
```

```
[assetExportSession setTimeRange:timeRange];
```

# AV Kit Wrap Up

- UI level Cocoa framework for AV Foundation
- Standard playback and trim controls through `AVPlayerView`
- Power of AV Foundation without custom view











# A Travel Guide to AV Foundation and AV Kit

Sam Bushell

# Do You Know the Way to AV Foundation?

- Depends how you use QuickTime
  - An easy change for some developers
  - Deeper refactoring needed for others
- Not an API-for-API swap
- Recent QTKit APIs are similar to their AV Foundation counterparts

# API Areas

- AV Resources
- Playback
- Exporting media files
- Reading and writing media files
- Retrieving frames during playback and offline
- Editing
- Metadata
- Capture
- Time and media samples
- Video compression and decompression



# Basics and Playback

# Representing AV Resources

QuickTime	QTKit	AV Foundation
Movie Track / Media	QTMovie QTTrack / QTMedia	AVAsset AVAssetTrack

# Creating Media Objects

QuickTime	QTKit	AV Foundation
NewMovieFromFile NewMovieFromDataRef NewMovieFromProperties (etc.)	QTMovie +[movieWithFile:error:] +[movieWithURL:error:] (etc.)	+[AVAsset assetWithURL:]

# Creating Media Objects

QuickTime	QTKit	AV Foundation
NewMovieFromFile NewMovieFromDataRef NewMovieFromProperties (etc.)	QTMovie +[movieWithFile:error:] +[movieWithURL:error:] (etc.)	+[AVAsset assetWithURL:]

```
AVAsset *asset = [AVAsset assetWithURL:...];  
[asset loadValuesAsynchronouslyForKeys:@[@"tracks",@"duration"]  
      completionHandler:^(...)];
```

# Playback

QuickTime	QTKit	AV Foundation
SetMovieRate SetMovieTime	QTMovie <ul style="list-style-type: none"><li>– play</li><li>– stop</li><li>– setCurrentTime:</li><li>– stepForward</li><li>– stepBackward</li></ul>	AVPlayer / AVQueuePlayer <ul style="list-style-type: none"><li>– play</li><li>– pause</li><li>rate <i>property</i></li></ul> AVPlayerItem <ul style="list-style-type: none"><li>– seekToTime: <i>and variants</i></li><li>– stepByCount:</li></ul>

# Playback

QuickTime	QTKit	AV Foundation
SetMovieRate SetMovieTime	QTMovie <ul style="list-style-type: none"><li>– play</li><li>– stop</li><li>– setCurrentTime:</li><li>– stepForward</li><li>– stepBackward</li></ul>	AVPlayer / AVQueuePlayer <ul style="list-style-type: none"><li>– play</li><li>– pause</li><li>rate <i>property</i></li></ul> AVPlayerItem <ul style="list-style-type: none"><li>– seekToTime: <i>and variants</i></li><li>– stepByCount:</li></ul>

```
[item seekToTime:t toleranceBefore:b toleranceAfter:a completionHandler:^(...)];
```



# Playback in AppKit NSViews



QTKit	AV Kit
<div>QTMovieView</div> <div><div>- play:</div><div>- pause:</div></div>	<div>AVPlayerView</div> <div>controlsStyle <i>property</i></div> <div>player <i>property</i></div> <div>AVPlayer</div> <div><div>- play</div><div>- pause</div><div>rate <i>property</i></div></div>

# Playback via Core Animation

QTKit	AV Foundation
<div>QTMovieLayer</div> <div>– setMovie:</div>	<div>AVPlayerLayer</div> <div>player <i>property</i></div> <div>AVSynchronizedLayer</div> <div>playerItem <i>property</i></div>

# Playback via Core Animation

QTKit	AV Foundation
<div>QTMovieLayer</div> <div>– setMovie:</div>	<div>AVPlayerLayer</div> <div>player <i>property</i></div> <div>AVSynchronizedLayer</div> <div>playerItem <i>property</i></div>



# Authoring and Editing

# Media Export

QuickTime	ConvertMovieToDataRef MovieExportToDataRef
QTKit	QTMovie – writeToFile:withAttributes: QTExportSession – initWithMovie:exportOptions:outputURL:error:
AV Foundation	AVAssetExportSession – initWithAsset:presetName: AVAssetReader <i>and</i> AVAssetWriter

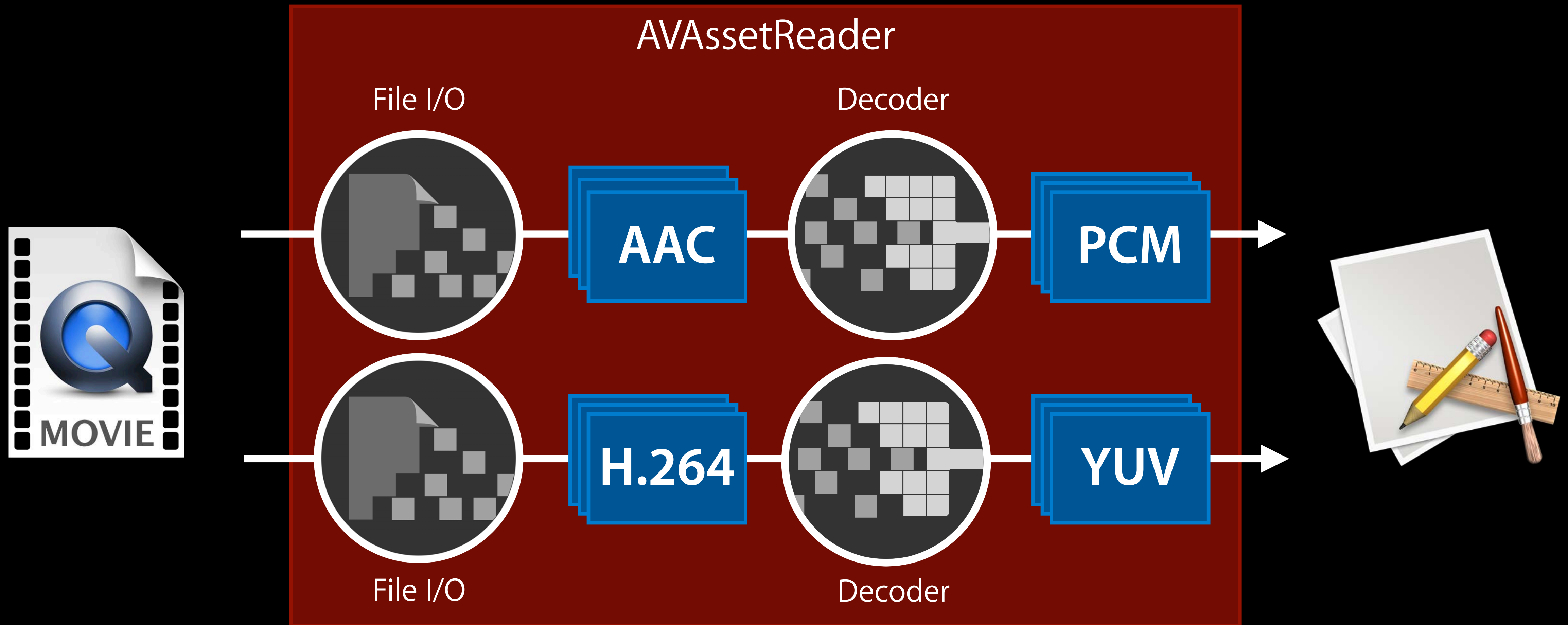
# Reading Media Files

QuickTime	AV Foundation
GetMediaSample2	AVAssetReader AVAssetReaderOutput ( <i>per-track</i> ) ( <i>created with</i> outputSettings:nil)
SetMovieGWorld / SetMovieVisualContext SetMovieTime MoviesTask  MovieAudioExtraction	AVAssetReader AVAssetReaderOutput ( <i>per-track</i> ) ( <i>non-nil</i> outputSettings)



# AVAssetReader

Deeply multithreaded



# Writing Media Files

QuickTime	AddMediaSample2 ... AddMovieToStorage MovieExportFromProceduresToDataRef
QTKit	QTMovie – initWithWritableDataReference:error: – addImage:forDuration:withAttributes:
AV Foundation	AVAssetWriter AVAssetWriterInput ( <i>per-track</i> ) AVOutputSettingsAssistant

# Writing Media Files



QuickTime	AddMediaSample2 ... AddMovieToStorage MovieExportFromProceduresToDataRef
QTKit	QTMovie – initWithWritableDataReference:error: – addImage:forDuration:withAttributes:
AV Foundation	AVAssetWriter AVAssetWriterInput ( <i>per-track</i> ) AVOutputSettingsAssistant



# Writing Media Files



QuickTime	AddMediaSample2 ... AddMovieToStorage MovieExportFromProceduresToDataRef
QTKit	QTMovie – initWithWritableDataReference:error: – addImage:forDuration:withAttributes:
AV Foundation	AVAssetWriter AVAssetWriterInput ( <i>per-track</i> ) AVOutputSettingsAssistant

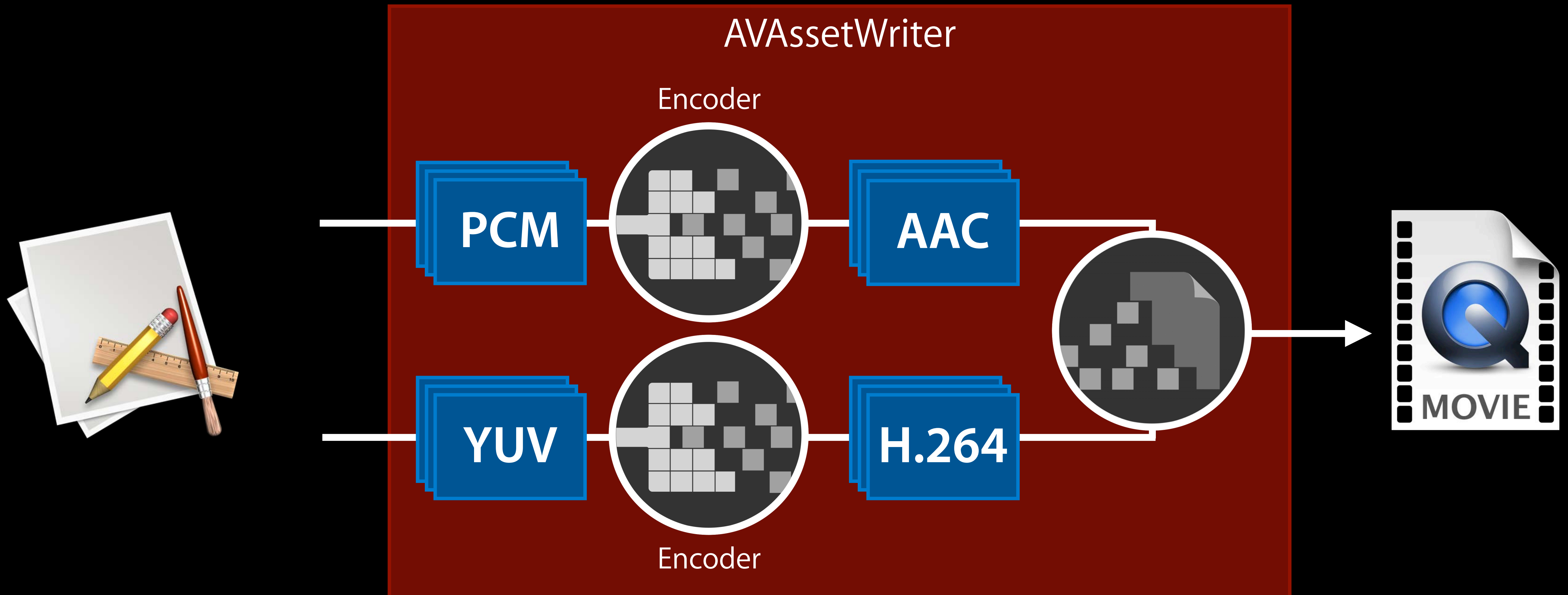
# Writing Media Files



QuickTime	AddMediaSample2 ... AddMovieToStorage MovieExportFromProceduresToDataRef
QTKit	QTMovie – initWithWritableDataReference:error: – addImage:forDuration:withAttributes:
AV Foundation	AVAssetWriter AVAssetWriterInput ( <i>per-track</i> )  AVOutputSettingsAssistant

# AVAssetWriter

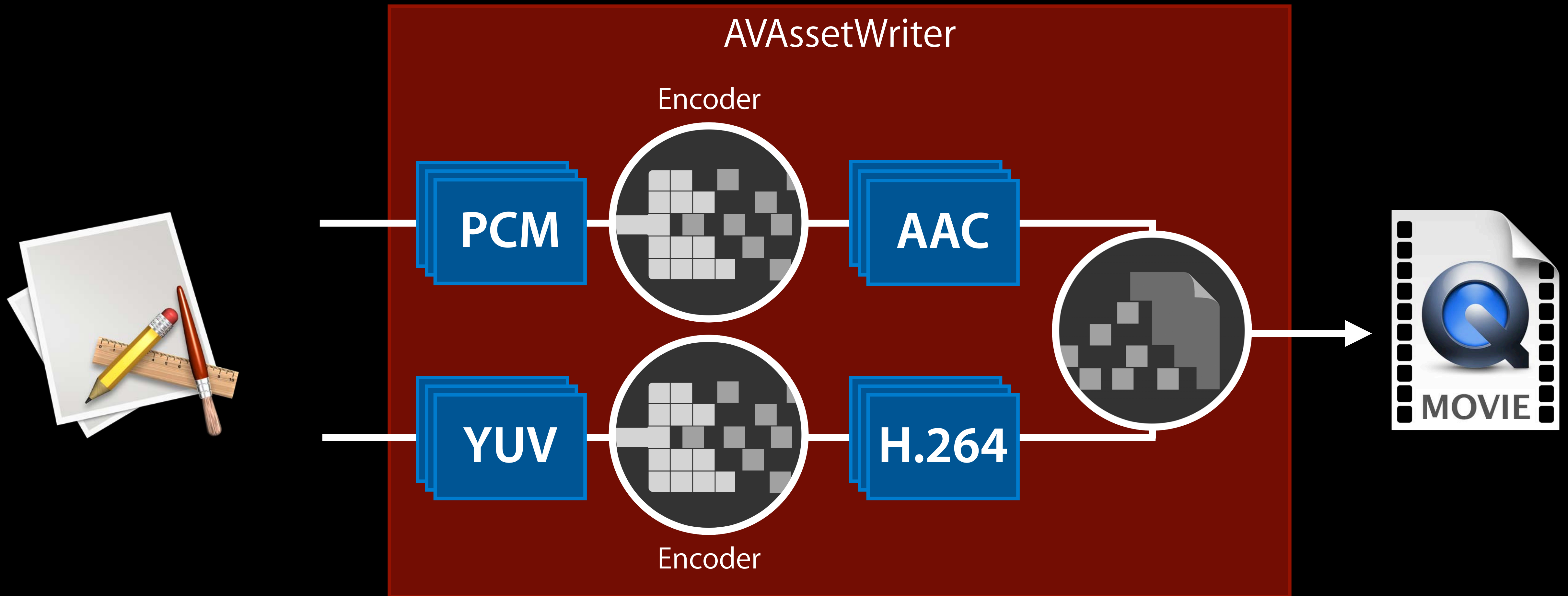
Deeply multithreaded





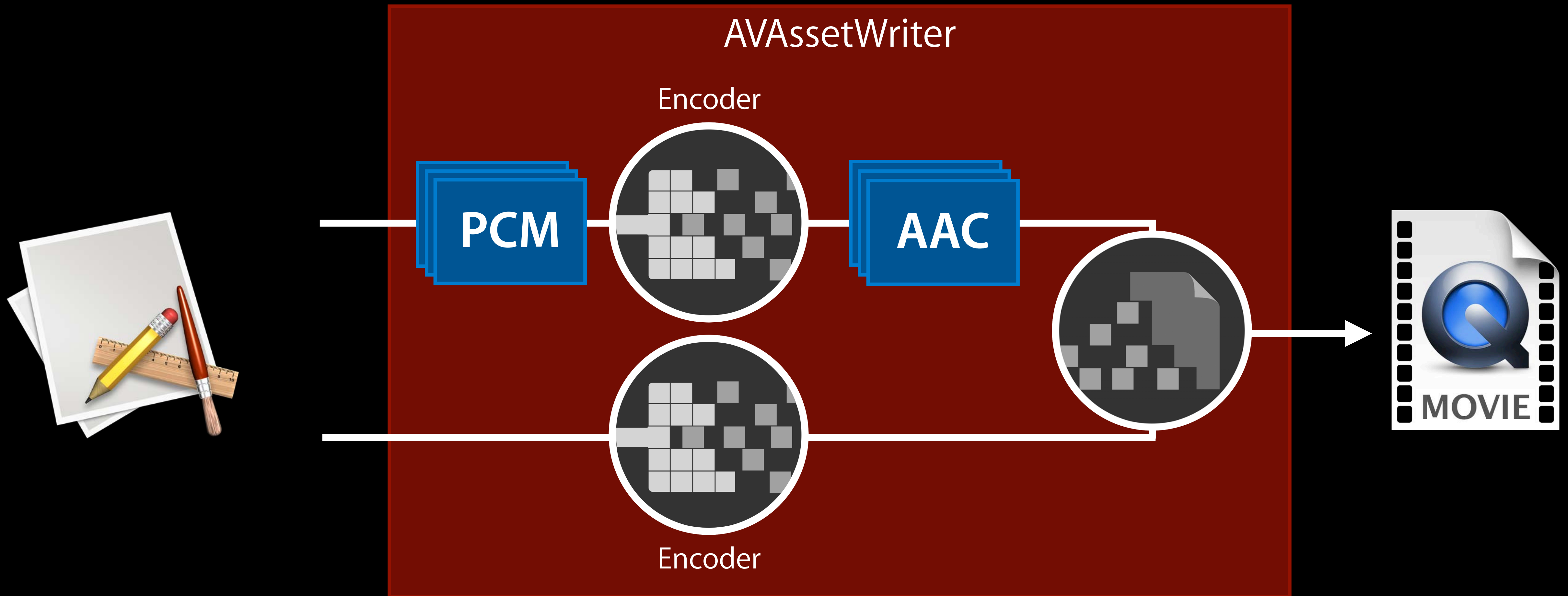
# AVAssetWriter

Deeply multithreaded



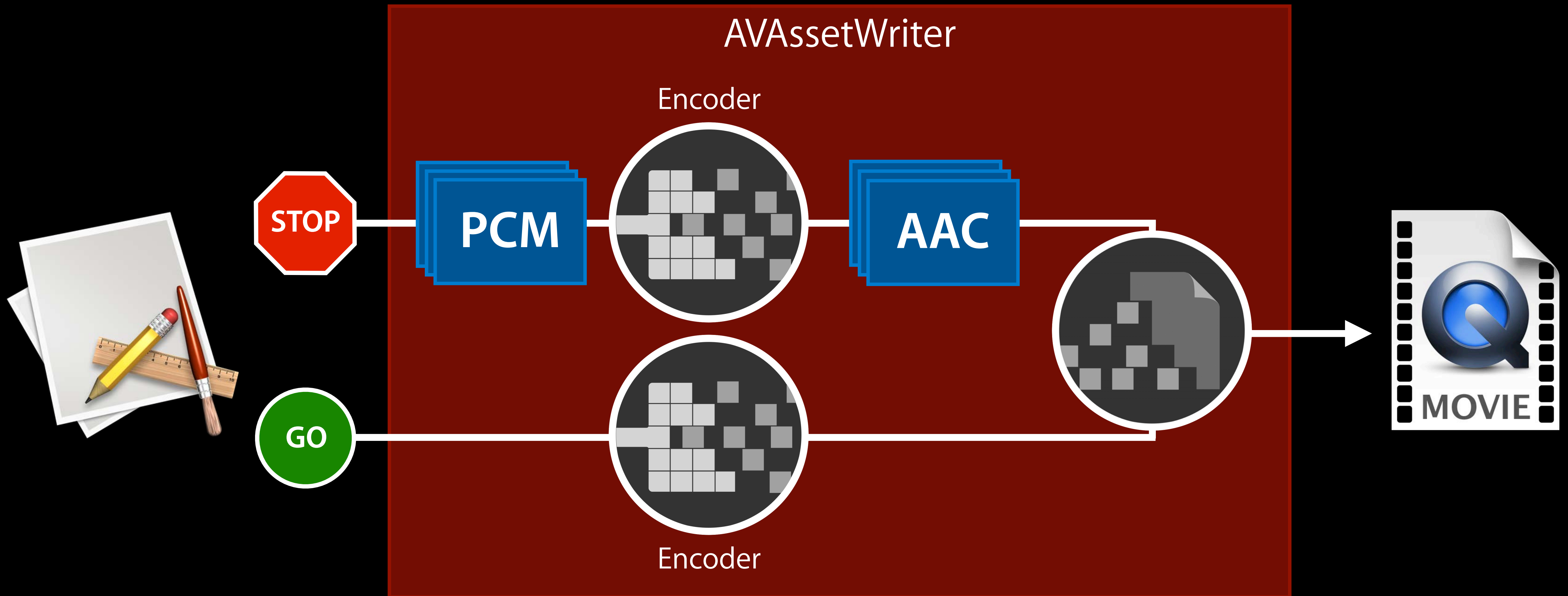
# AVAssetWriter

Deeply multithreaded



# AVAssetWriter

Deeply multithreaded





# Getting Video Frames During Playback

QuickTime	SetMovieVisualContext
QTKit	QTMovie – setVisualContext:
AV Foundation	AVPlayerItemVideoOutput

# Accessing Audio During Playback



QuickTime	QTAudioContextRegisterInsert SetMovieAudioContext
QTKit	—
AV Foundation	MTAudioProcessingTap AVAudioMix.audioTapProcessor <i>property</i>

# Getting Still Images from Video

QTKit	QTMovie – frameImageAtTime:
AV Foundation	AVAssetImageGenerator – copyCGImageAtTime:actualTime:error: – generateCGImagesAsynchronouslyForTimes: completionHandler:

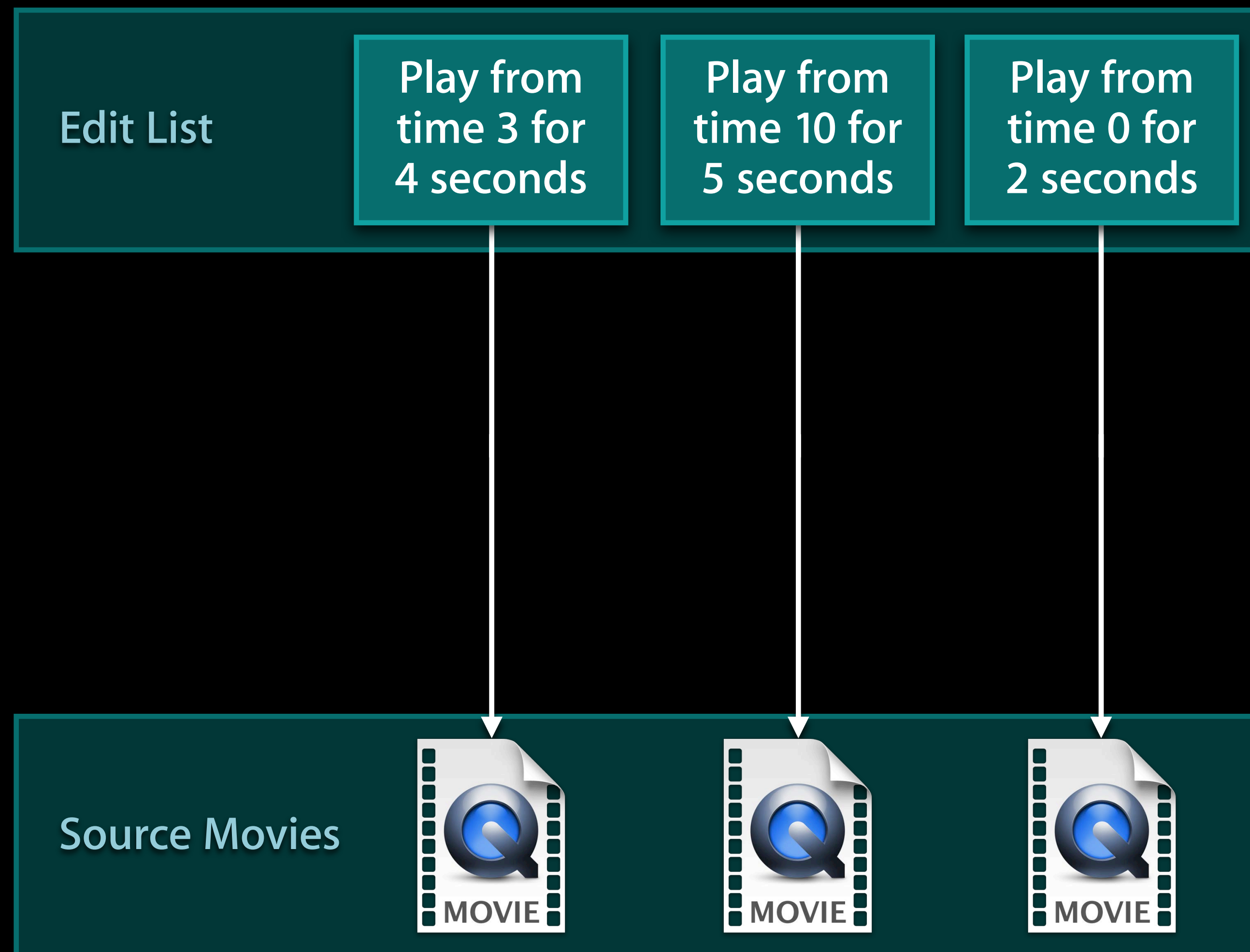


# Editing

<b>QuickTime</b>	InsertMovieSegment DeleteMovieSegment ScaleMovieSegment	InsertTrackSegment DeleteTrackSegment ScaleTrackSegment
<b>QTKit</b>	QTMovie – insertSegmentOfMovie:... – deleteSegment: – scaleSegment:newDuration:	QTTrack – insertSegmentOfTrack:... – deleteSegment: – scaleSegment:newDuration:
<b>AV Foundation</b>	AVMutableComposition – insertTimeRange:ofAsset:... – removeTimeRange: – scaleTimeRange:toDuration:	AVMutableCompositionTrack – insertTimeRange:ofTrack:... – removeTimeRange: – scaleTimeRange:toDuration:

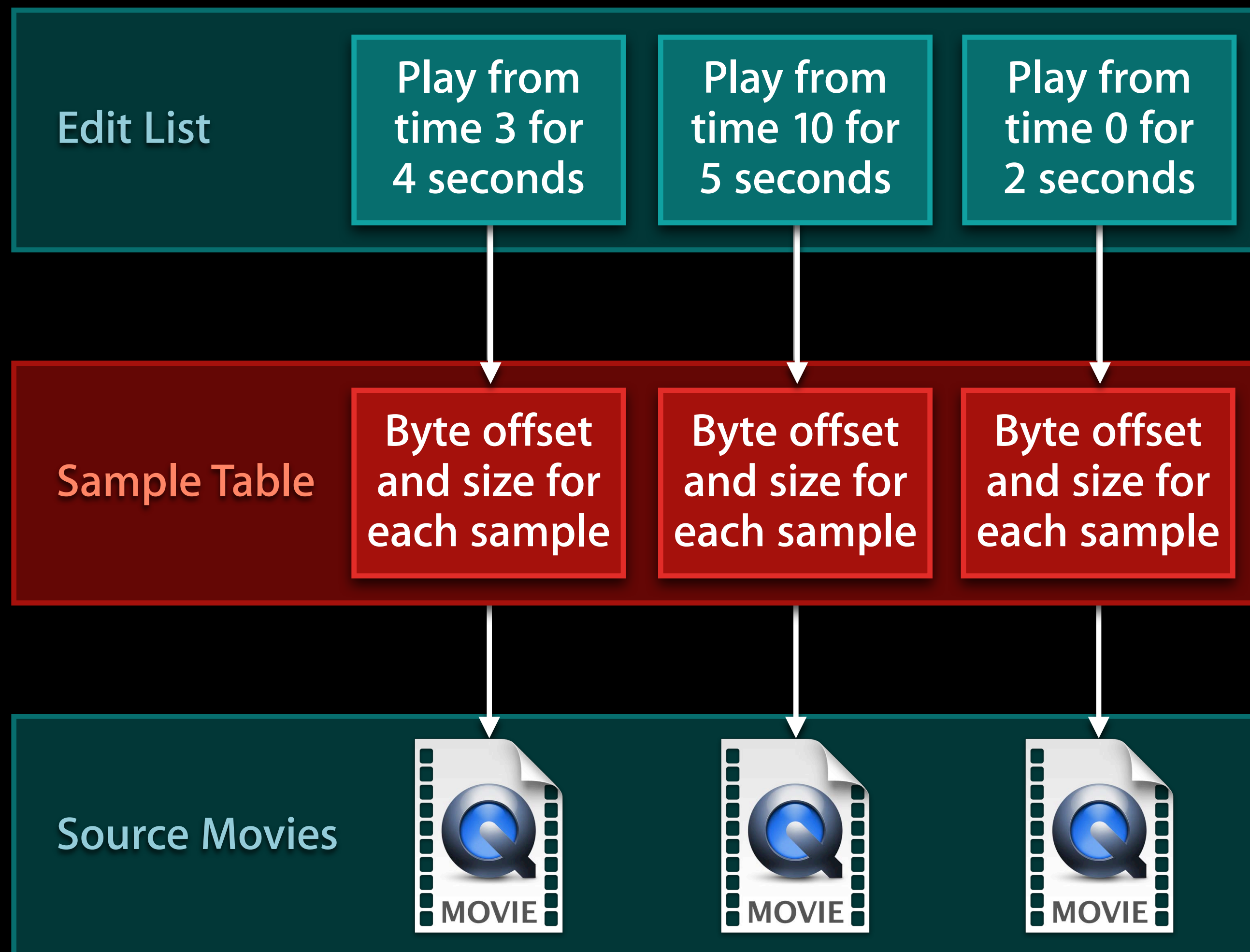
# Editing

## Edit lists and sample tables



# Editing

## Edit lists and sample tables

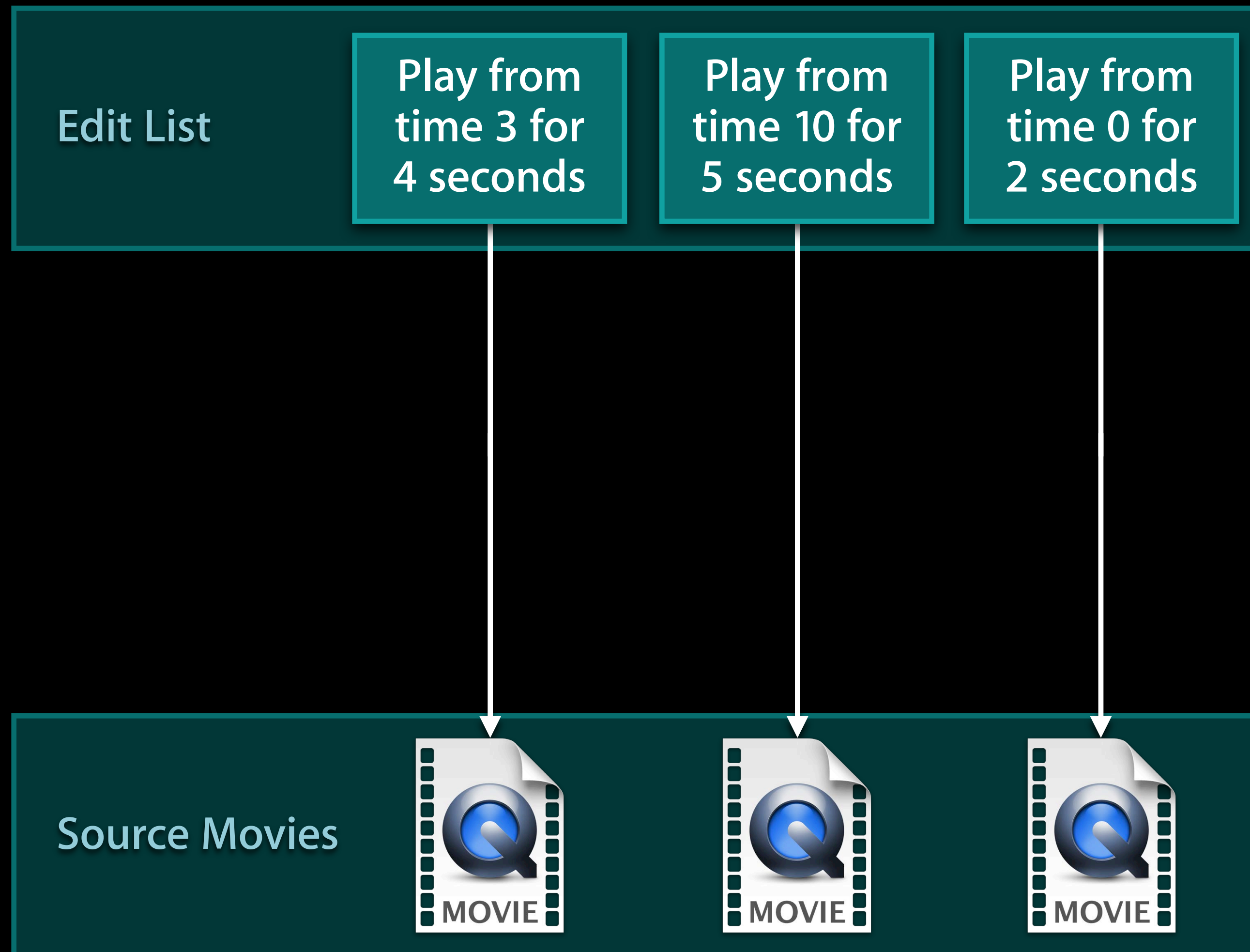


Required by  
QuickTime API



# Editing

## Edit lists and sample tables



Not required by  
AV Foundation

# Metadata

QTKit	AV Foundation
<div>QTMovie</div> <ul style="list-style-type: none"><li>– commonMetadata</li><li>– availableMetadataFormats</li><li>– metadataForFormat:</li></ul> <div>QTMetadataItem</div> <div><i>properties:</i> key, keySpace, locale, time, value, extraAttributes (etc.)</div>	<div>AVAsset</div> <div>commonMetadata <i>property</i></div> <div>availableMetadataFormats <i>property</i></div> <ul style="list-style-type: none"><li>– metadataForFormat:</li></ul> <div>AVMetadataItem</div> <div><i>properties:</i> key, keySpace, locale, time, value, extraAttributes (etc.)</div>

# Metadata

QTKit	AV Foundation
<p>QTMovie</p> <ul style="list-style-type: none"><li>– commonMetadata</li><li>– availableMetadataFormats</li><li>– metadataForFormat:</li></ul> <p>QTMetadataItem</p> <p><i>properties:</i> key, keySpace, locale, time, value, extraAttributes (etc.)</p>	<p>AVAsset</p> <p>commonMetadata <i>property</i></p> <p>availableMetadataFormats <i>property</i></p> <ul style="list-style-type: none"><li>– metadataForFormat:</li></ul> <p>AVMetadataItem</p> <p><i>properties:</i> key, keySpace, locale, time, value, extraAttributes (etc.)</p>

```
[asset loadValuesAsynchronouslyForKeys:@[@"commonMetadata"]  
      completionHandler:^(...)];
```



# Capture

# Capture

QuickTime	QTKit	AV Foundation
Sequence Grabber	QTCaptureSession QTCaptureInput QTCaptureOutput QTCaptureConnection	AVCaptureSession AVCaptureInput AVCaptureOutput AVCaptureConnection

# Capture Connections

Between inputs and outputs



# Capture Connections

## Between inputs and outputs

Input



# Capture Connections

Between inputs and outputs

Input



Preview



# Capture Connections

Between inputs and outputs

Input



Preview



Output





# Capture Connections

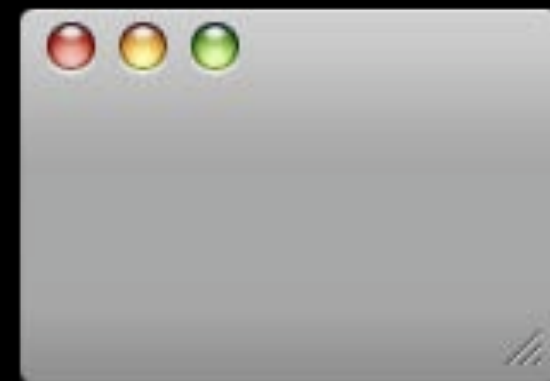
## Between inputs and outputs

Video connection —

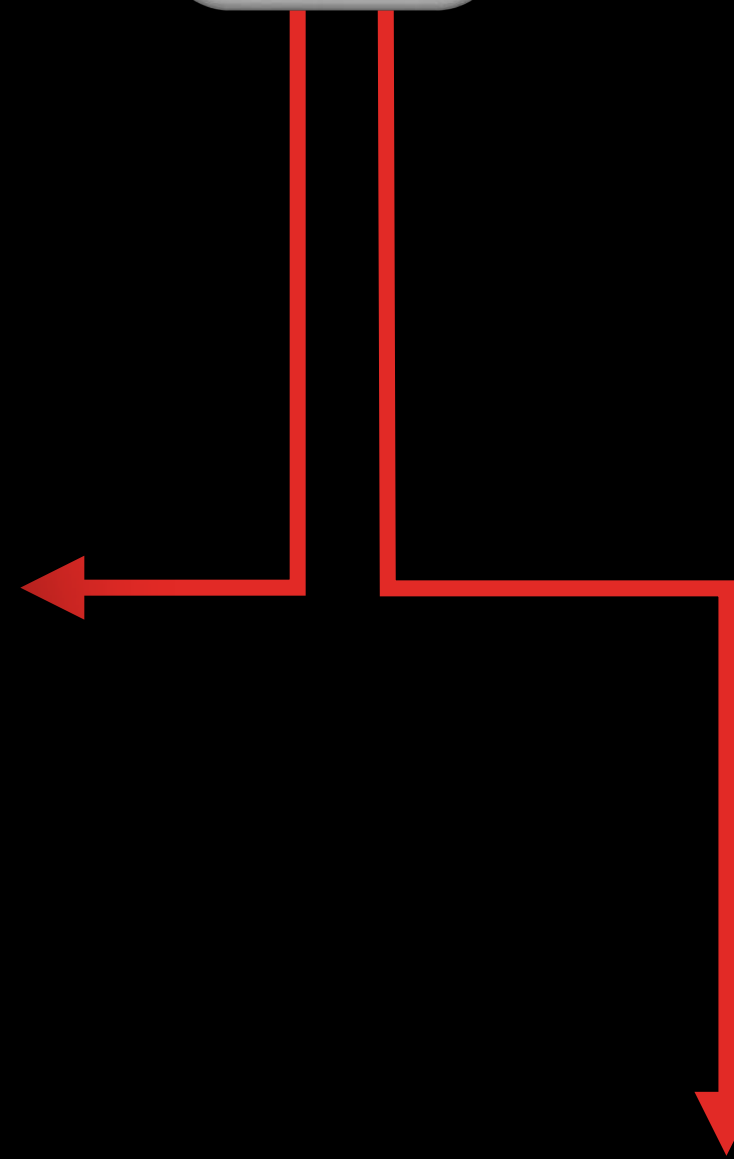
Input



Preview



Output



# Capture Connections

## Between inputs and outputs

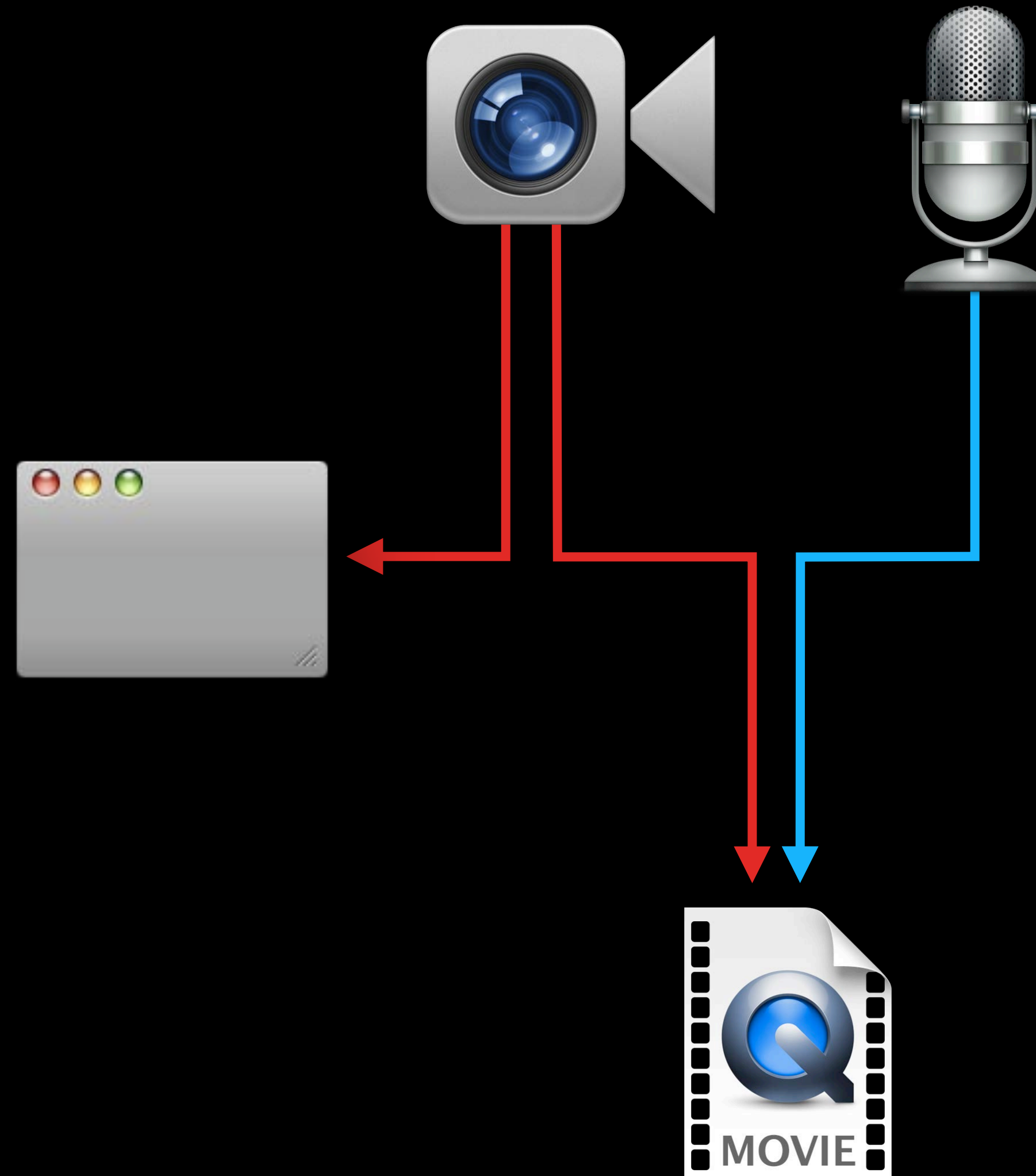
Video connection —

Audio connection —

Input

Preview

Output



# Capture Connections

## Between inputs and outputs

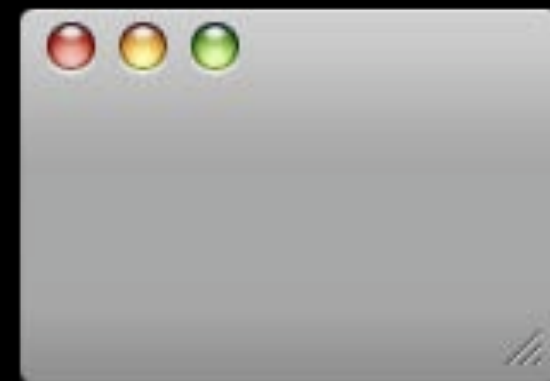
Video connection —

Audio connection —

Input



Preview



Output





# Capture Connections

## Between inputs and outputs

Video connection —

Audio connection —

Input



Preview



Output



# Capture Connections

## Between inputs and outputs

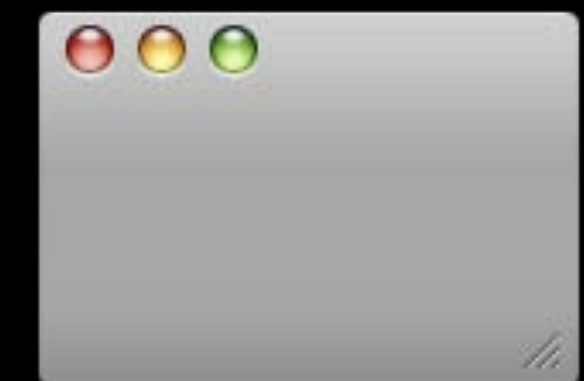
Video connection —

Audio connection —

Input



Preview



Output



# Capture Connections

Between inputs and outputs

Video connection —

Audio connection —

Input



Preview



Output





# Capture Connections

Between inputs and outputs

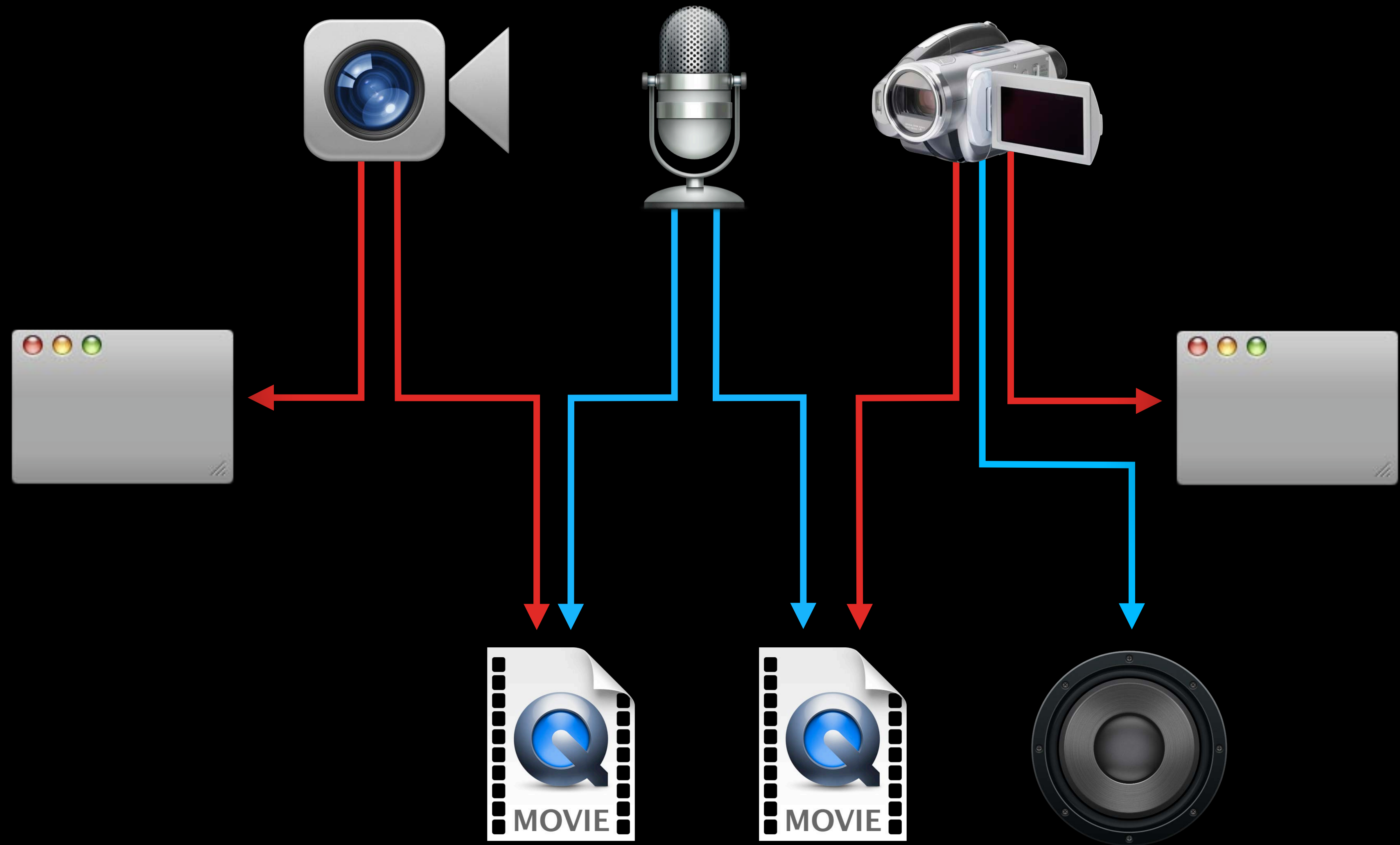
Video connection —

Audio connection —

Input

Preview

Output



# Input/Output

QTKit	AV Foundation
QTCaptureDeviceInput	AVCaptureDeviceInput AVCaptureScreenInput
QTCaptureAudioPreviewOutput QTCaptureVideoPreviewOutput QTCaptureDecompressedAudioOutput QTCaptureDecompressedVideoOutput QTCaptureMovieFileOutput	AVCaptureAudioPreviewOutput  AVCaptureAudioDataOutput AVCaptureVideoDataOutput AVCaptureMovieFileOutput AVCaptureAudioFileOutput AVCaptureStillImageOutput

# Input/Output

QTKit	AV Foundation
QTCaptureDeviceInput	AVCaptureDeviceInput AVCaptureScreenInput
QTCaptureAudioPreviewOutput QTCaptureVideoPreviewOutput QTCaptureDecompressedAudioOutput QTCaptureDecompressedVideoOutput QTCaptureMovieFileOutput	AVCaptureAudioPreviewOutput  AVCaptureAudioDataOutput AVCaptureVideoDataOutput AVCaptureMovieFileOutput AVCaptureAudioFileOutput AVCaptureStillImageOutput



# Input/Output

QTKit	AV Foundation
QTCaptureDeviceInput	AVCaptureDeviceInput AVCaptureScreenInput
QTCaptureAudioPreviewOutput QTCaptureVideoPreviewOutput QTCaptureDecompressedAudioOutput QTCaptureDecompressedVideoOutput QTCaptureMovieFileOutput	AVCaptureAudioPreviewOutput  AVCaptureAudioDataOutput AVCaptureVideoDataOutput AVCaptureMovieFileOutput AVCaptureAudioFileOutput AVCaptureStillImageOutput

# Input/Output

QTKit	AV Foundation
QTCaptureDeviceInput	AVCaptureDeviceInput AVCaptureScreenInput
QTCaptureAudioPreviewOutput QTCaptureVideoPreviewOutput QTCaptureDecompressedAudioOutput QTCaptureDecompressedVideoOutput QTCaptureMovieFileOutput	AVCaptureAudioPreviewOutput  AVCaptureAudioDataOutput AVCaptureVideoDataOutput AVCaptureMovieFileOutput AVCaptureAudioFileOutput AVCaptureStillImageOutput

# Input/Output

QTKit	AV Foundation
QTCaptureDeviceInput	AVCaptureDeviceInput AVCaptureScreenInput
QTCaptureAudioPreviewOutput QTCaptureVideoPreviewOutput QTCaptureDecompressedAudioOutput QTCaptureDecompressedVideoOutput QTCaptureMovieFileOutput	AVCaptureAudioPreviewOutput  AVCaptureAudioDataOutput AVCaptureVideoDataOutput AVCaptureMovieFileOutput AVCaptureAudioFileOutput AVCaptureStillImageOutput



# Input/Output

QTKit	AV Foundation
QTCaptureDeviceInput	AVCaptureDeviceInput AVCaptureScreenInput
QTCaptureAudioPreviewOutput QTCaptureVideoPreviewOutput QTCaptureDecompressedAudioOutput QTCaptureDecompressedVideoOutput QTCaptureMovieFileOutput	AVCaptureAudioPreviewOutput  AVCaptureAudioDataOutput AVCaptureVideoDataOutput AVCaptureMovieFileOutput AVCaptureAudioFileOutput AVCaptureStillImageOutput

# Device Access

QTKit	AV Foundation
<div>QTCaptureDevice</div> <ul style="list-style-type: none"><li>+ inputDevices</li><li>+ inputDevicesWithMediaType:</li><li>+ defaultInputDeviceWithMediaType:</li><li>+ deviceWithUniqueID:</li></ul>	<div>AVCaptureDevice</div> <ul style="list-style-type: none"><li>+ devices</li><li>+ devicesWithMediaType:</li><li>+ defaultDeviceWithMediaType:</li><li>+ deviceWithUniqueID:</li></ul>

# Display of What's Being Recorded

QTKit	AV Foundation
QTCaptureLayer QTCaptureView	AVCaptureVideoPreviewLayer



# Low-Level Media Objects

# Representation of Time Values

64-bit time value (numerator)

---

32-bit time scale (denominator)

# Representation of Time Values

64-bit time value (numerator)

32-bit time scale (denominator)

QuickTime	QTKit	AV Foundation and Core Media
TimeRecord (64-bit) TimeValue (32-bit)	QTTime QTTimeRange	CMTIME CMTimeRange CMTimeMapping



# Representation of Moving Time

QuickTime	AV Foundation and Core Media
Clock TimeBase	CMClock CMTimebase

# Describing Compressed Media Samples

QuickTime	QTKit	AV Foundation and Core Media
SampleDescriptionHandle  ImageDescriptionHandle SoundDescriptionHandle <i>(etc.)</i>	QTFormatDescription	CMFormatDescription  CMVideoFormatDescription CMAudioFormatDescription <i>(etc.)</i>
	QTSampleBuffer	CMSampleBuffer

# Video Compression and Decompression

QuickTime's Image Compression Manager	AV Foundation's Video Toolbox
<code>CompressSequenceBegin</code> <code>DecompressSequenceBegin</code>  <code>ICMCompressionSession</code> <code>ICMDecompressionSession</code>	<code>VTCompressionSession</code> <code>VTDecompressionSession</code> <code>VTPixelFormatTransferSession</code>



# Still Image I/O

QuickTime	ImageIO
GraphicsImporter GraphicsExporter	CGImageSource CGImageDestination

AVAsset, AVAssetTrack

AVPlayer, AVPlayerItem, AVPlayerItemTrack

AVQueuePlayer, AVPlayerLayer, AVPlayerView

AVSynchronizedLayer

AVAssetExportSession

AVAssetReader, AVAssetReaderInput

AVAssetWriter, AVAssetWriterOutput

AVOutputSettingsAssistant

AVPlayerItemVideoOutput

AVAudioMix, MTAudioProcessingTap

AVAssetImageGenerator

AVComposition, AVCompositionTrack

AVMetadataItem

AVCaptureSession, AVCaptureConnection

AVCaptureDeviceInput, AVCaptureScreenInput

AVCaptureAudioPreviewOutput,

AVCaptureAudioDataOutput,

AVCaptureVideoDataOutput,

AVCaptureMovieFileOutput,

AVCaptureAudioFileOutput,

AVCaptureStillImageOutput

AVCaptureVideoPreviewLayer

CMTime, CMTimeRange, CMTimeMapping

CMClock, CMTimebase

CMFormatDescription,

CMVideoFormatDescription,

CMAudioFormatDescription

CMSampleBuffer, CMBlockBuffer

VTCompressionSession,

VTDecompressionSession,

VTPixelFormatSession

# Summary

- QuickTime.framework and QTKit.framework APIs deprecated in OS X 10.9
  - Reminding you to make the transition
  - Your apps will still run
- AV Foundation and AV Kit are our modern media frameworks
  - And they're better!
- QuickTime Movie file format is still our primary file format
- QTMovieModernizer
- AVPlayerView

We welcome  
your feedback



# More Information

## John Geleynse

Director, Technology Evangelism  
[geleynse@apple.com](mailto:geleynse@apple.com)

## Documentation

AV Foundation Programming Guide

<https://developer.apple.com/library/mac/#documentation/AudioVideo/Conceptual/AVFoundationPG/>

Technical Note TN2300: Transitioning QTKit Code to AV Foundation

<https://developer.apple.com/library/mac/technotes/tn2300/>

## Apple Developer Forums

<http://devforums.apple.com>

# Related Sessions

Preparing and Presenting Media for Accessibility

Nob Hill  
Wednesday 10:15AM

What's New in Camera Capture

Nob Hill  
Wednesday 11:30AM

Advanced Editing with AV Foundation

Marina  
Thursday 9:00AM

# Labs

OS X and iOS Capture Lab

Media Lab B  
Thursday 9:00AM

AV Foundation Lab

Media Lab B  
Thursday 2:00PM

AV Foundation Lab

Media Lab B  
Friday 9:00AM

 WWDC2013