HEVC Video with Alpha

Shiva Sundar, CoreMedia Engineering
Alpha Channels

- 0% opaque (fully transparent)
- 100% opaque
- 50% opaque
Alpha Channels

- 0% opaque (fully transparent)
- 100% opaque
- 50% opaque
Formats with Alpha Channels

Still Image

High bitrate Lossless/Mezzanine

PNG, TIFF
## Formats with Alpha Channels

<table>
<thead>
<tr>
<th>High bitrate Lossless/Mezzanine</th>
<th>Still Image</th>
<th>Motion Video</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PNG, TIFF</td>
<td>Apple ProRes 4444</td>
</tr>
<tr>
<td>Formats with Alpha Channels</td>
<td>Still Image</td>
<td>Motion Video</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>High bitrate</td>
<td>PNG, TIFF</td>
<td>Apple ProRes 4444</td>
</tr>
<tr>
<td>Lossless/Mezzanine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flexible bitrate</td>
<td>HEIF</td>
<td></td>
</tr>
</tbody>
</table>
## Formats with Alpha Channels

<table>
<thead>
<tr>
<th></th>
<th>Still Image</th>
<th>Motion Video</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High bitrate</strong></td>
<td>PNG, TIFF</td>
<td>Apple ProRes 4444</td>
</tr>
<tr>
<td><strong>Lossless/Mezzanine</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Flexible bitrate</strong></td>
<td>HEIF / HEIFS</td>
<td></td>
</tr>
</tbody>
</table>
# Formats with Alpha Channels

<table>
<thead>
<tr>
<th></th>
<th>Still Image</th>
<th>Motion Video</th>
</tr>
</thead>
<tbody>
<tr>
<td>High bitrate</td>
<td>PNG, TIFF</td>
<td>Apple ProRes 4444</td>
</tr>
<tr>
<td>Lossless/Mezzanine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flexible bitrate</td>
<td>HEIF / HEIFS</td>
<td>HEVC with Alpha</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Platforms

iOS 13

tvOS 13

macOS Catalina
Use Cases
Use Cases
Use Cases

Video Overlays
Use Cases

Video Overlays
Use Cases

Video Overlays
Use Cases

Video Overlays

SpriteKit / SceneKit / Metal
Use Cases

Video Overlays

SpriteKit / SceneKit / Metal

Safari
Use Cases

Video Overlays

SpriteKit / SceneKit / Metal

Safari
Use Cases

Video Overlays
SpriteKit / SceneKit / Metal
Safari
Background Removal
Use Cases
Use Cases
Use Cases
Use Cases
Use Cases
How it works
How it works
How it works

![Diagram showing how Base and Alpha layers combine to create a MOV file.](image)
How it works
How it works
Encoding
Encoding

Video Frames with Alpha → AVAssetWriter → HEVC with Alpha
Encoding

Any Video with Alpha -> AVAssetExportSession -> HEVC with Alpha

AVAssetExportPresetHEVC...WithAlpha
Encoding

Video Frames with Alpha

VTCompressionSession

Sample Buffers

Base

Alpha

Base

Alpha

Base

Alpha
Playback
Playback

HEVC with Alpha

Displayed with Transparent Background
Playback

HEVC with Alpha

Displayed with Transparent Background
Playback

HEVC with Alpha

AVPlayer

AVPlayerItemVideoOutput

Video Frames with Alpha
Decoding
Decoding

HEVC with Alpha → AVAssetImageGenerator → CGImage
Decoding HEVC with Alpha Video Frames with Alpha

MOV

AVAssetReader

HEVC with Alpha

Video Frames with Alpha
Decoding

Sample Buffers

VTDecompressionSession

Video Frames with Alpha
### APIs

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Playback</th>
<th>Decoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVAssetWriter</td>
<td>AVPlayer</td>
<td>AVAssetImageGenerator</td>
</tr>
<tr>
<td>AVAssetExportSession</td>
<td>AVPlayerItemVideoOutput</td>
<td>AVAssetReader</td>
</tr>
<tr>
<td>VTCompressionSession</td>
<td></td>
<td>VTDecompressionSession</td>
</tr>
</tbody>
</table>
For Example
// AVAssetWriterInput - CodecType

import AVFoundation

let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height] as [String: Any]

let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
// AVAssetWriterInput - CodecType

import AVFoundation

let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height] as [String: Any]

let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
Premultiplied Alpha

Premultiplied Alpha

Straight Alpha
Premultiplied Alpha

Both Premultiplied Alpha and Straight Alpha are supported
Both Premultiplied Alpha and Straight Alpha are supported

Premultiplied Alpha is recommended
// AVAssetWriterInput
import AVFoundation
import VideoToolbox

let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height
    kVTCompressionPropertyKey_AlphaChannelMode: kVTAlphaChannelMode_PremultipliedAlpha
] as [String: Any]

let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)

CVBufferSetAttachment(
    pixelBuffer,
    CVImageBufferAlphaChannelModeKey,
    kCVImageBufferAlphaChannelMode_PremultipliedAlpha,
    .shouldPropagate)
// AVAssetWriterInput
import AVFoundation
import VideoToolbox
let outputSettings = [
  AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
  AVVideoWidthKey: width,
  AVVideoHeightKey: height
  kVTCompressionPropertyKey_AlphaChannelMode: kVTAlphaChannelMode_PremultipliedAlpha]
  as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)

CVBufferSetAttachment(
  pixelBuffer,
  CVImageBufferAlphaChannelModeKey,
  kCVImageBufferAlphaChannelMode_PremultipliedAlpha,
  .shouldPropagate)
import AVFoundation
import VideoToolbox

let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height
    kVTCompressionPropertyKey_AlphaChannelMode:
    kVTAlphaChannelMode_PremultipliedAlpha
] as [String: Any]

let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)

CVBufferSetAttachment(
    pixelBuffer,
    CVImageBufferAlphaChannelModeKey,
    kCVImageBufferAlphaChannelMode_PremultipliedAlpha,
    .shouldPropagate)
Bitrate and Quality

Can control bitrate of base layer (bits per second)

Can control quality of alpha layer (0 to 1)
import AVFoundation
import VideoToolbox

let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height,
    AVVideoCompressionPropertiesKey: [
        kVTCompressionPropertyKey_AverageBitRate: baseLayerBitrate,
        kVTCompressionPropertyKey_TargetQualityForAlpha: alphaLayerQuality
    ] as [String: Any]
]

let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
// AVAssetWriterInput - BitRate and Quality tradeoffs

import AVFoundation
import VideoToolbox

let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height,
    AVVideoCompressionPropertiesKey:
        [kVTCompressionPropertyKey_AverageBitRate: baseLayerBitrate,
         kVTCompressionPropertyKey_TargetQualityForAlpha: alphaLayerQuality]
] as [String: Any]

let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
// Presence of Alpha Channel

import AVFoundation

let hevcWithAlphaTracks = asset.tracks(withMediaCharacteristic:.containsAlphaChannel)

let containsAlphaChannel = formatDescription.extensions[.containsAlphaChannel] as! Bool?
// Presence of Alpha Channel

import AVFoundation

let hevcWithAlphaTracks = asset.tracks(withMediaCharacteristic:.containsAlphaChannel)

let containsAlphaChannel = formatDescription.extensions[.containsAlphaChannel] as! Bool?
// Presence of Alpha Channel
import AVFoundation
let hevcWithAlphaTracks = asset.tracks(withMediaCharacteristic:.containsAlphaChannel)

let containsAlphaChannel = formatDescription.extensions[.containsAlphaChannel] as? Bool
// AVAssetExportSession - compatibility
import AVFoundation
AVAssetExportSession.determineCompatibility(
ofExportPreset: AVAssetExportPresetHEVCHighestQualityWithAlpha,
with: asset,
outputFileType: .mov) { compatible in
  if compatible {
    // Export
  } else {
    print("Export Session failed compatibility check")
    // Handle failure
  }
}
// AVAssetExportSession - compatibility
import AVFoundation

AVAssetExportSession.determineCompatibility(
ofExportPreset: AVAssetExportPresetHEVCHighestQualityWithAlpha,
with: asset,
outputFileType: .mov) { compatible in
  if compatible {
    // Export
  } else {
    print("Export Session failed compatibility check")
    // Handle failure
  }
}
// AVAssetExportSession - backwards compatibility
import AVFoundation

// Setup to use preferred color background
let prototypeInstruction = AVMutableVideoCompositionInstruction()
prototypeInstruction.backgroundColor = preferredColor // CGColor with ColorSpace

let videoComposition = AVMutableVideoComposition(propertiesOf: asset,
  prototypeInstruction: prototypeInstruction)

// Export
exportSession.outputURL = destinationURL
exportSession.outputFileType = .mov
exportSession.videoComposition = videoComposition
exportSession.exportAsynchronously {
  // Handle completion
}
// AVAssetExportSession - backwards compatibility

import AVFoundation

// Setup to use preferred color background

let prototypeInstruction = AVMutableVideoCompositionInstruction()
prototypeInstruction.backgroundColor = preferredColor // CGColor with ColorSpace

let videoComposition = AVMutableVideoComposition(propertiesOf: asset,
                                                  prototypeInstruction: prototypeInstruction)

// Export

exportSession.outputURL = destinationURL
exportSession.outputFileType = .mov
exportSession.videoComposition = videoComposition
exportSession.exportAsynchronously {
    // Handle completion
}
Interoperability Profile

Single video track with ‘hvc1’ codec type

• Two layers
  - Base layer: main profile; 4:2:0 video-range; nuh_layer_id 0
  - Alpha layer: main profile; 4:2:0 full-range, neutral chroma channel; nuh_layer_id 1
• Base and alpha layers share one VPS
  - VPS extension required — indicates presence of alpha channel
• Base and alpha layers have distinct PPS and SPS with different IDs
• Base and alpha layers must have identical frame types and dependency structure
• SEI alpha_channel_info message in hvcC in video format description
  - Indicates premultiplied alpha or straight alpha
HEVC Video with Alpha

Advanced lossy compression technology

Supported in iOS 13, tvOS 13, macOS Catalina

Hardware accelerated on recent devices

Integrated with media APIs

Integrated in Safari
More Information

developer.apple.com/wwdc19/506
HEVC Video with Alpha

Shiva Sundar, CoreMedia Engineering
Alpha Channels

- 0% opaque (fully transparent)
- 100% opaque
- 50% opaque
Alpha Channels

0% opaque (fully transparent)

100% opaque

50% opaque
Formats with Alpha Channels

Still Image

High bitrate
Lossless/Mezzanine

PNG, TIFF
## Formats with Alpha Channels

<table>
<thead>
<tr>
<th>High bitrate Lossless/Mezzanine</th>
<th>Still Image</th>
<th>Motion Video</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNG, TIFF</td>
<td>Apple ProRes 4444</td>
<td></td>
</tr>
</tbody>
</table>
# Formats with Alpha Channels

<table>
<thead>
<tr>
<th></th>
<th>Still Image</th>
<th>Motion Video</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High bitrate</strong></td>
<td>PNG, TIFF</td>
<td>Apple ProRes 4444</td>
</tr>
<tr>
<td>Lossless/Mezzanine</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Flexible bitrate</strong></td>
<td>HEIF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Formats with Alpha Channels

<table>
<thead>
<tr>
<th></th>
<th>Still Image</th>
<th>Motion Video</th>
</tr>
</thead>
<tbody>
<tr>
<td>High bitrate</td>
<td>PNG, TIFF</td>
<td>Apple ProRes 4444</td>
</tr>
<tr>
<td>Lossless/Mezzanine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flexible bitrate</td>
<td>HEIF / HEIFS</td>
<td></td>
</tr>
</tbody>
</table>
## Formats with Alpha Channels

<table>
<thead>
<tr>
<th></th>
<th>Still Image</th>
<th>Motion Video</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High bitrate</strong></td>
<td>PNG, TIFF</td>
<td>Apple ProRes 4444</td>
</tr>
<tr>
<td><strong>Lossless/Mezzanine</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Flexible bitrate</strong></td>
<td>HEIF / HEIFS</td>
<td>HEVC with Alpha</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Platforms

iOS 13

tvOS 13

macOS Catalina
Use Cases
Use Cases

Video Overlays
Use Cases

Video Overlays
Use Cases

Video Overlays
Use Cases

Video Overlays

SpriteKit / SceneKit / Metal
Use Cases

Video Overlays

SpriteKit / SceneKit / Metal

Safari
Use Cases

Video Overlays

SpriteKit / SceneKit / Metal

Safari
Use Cases

Video Overlays

SpriteKit / SceneKit / Metal

Safari

Background Removal
Use Cases
Use Cases
How it works
How it works
How it works

Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha
Base
Alpha

Base Layer

Alpha Layer
How it works
How it works

Base Layer

MOV file
Encoding
Encoding

Video Frames with Alpha → AVAssetWriter → HEVC with Alpha
Encoding

Any Video with Alpha

AVAssetExportSession

HEVC with Alpha

AVAssetExportPresetHEVCWithAlpha
Encoding

MOV

HEVC with Alpha

AVAssetExportSession

Video Composition on White Background

MOV

H.264 (No Alpha)

AVAssetExportPresetHighestQuality
Encoding

Video Frames with Alpha → VTCompressionSession → Sample Buffers

Base Alpha Base Alpha Base Alpha
Playback
Playback

HEVC with Alpha

Displayed with Transparent Background
Playback

HEVC with Alpha

Displayed with Transparent Background
Playback

MOV

HEVC with Alpha

AVPlayer

AVPlayerItemVideoOutput

Video Frames with Alpha
Decoding
Decoding HEVC with Alpha Video Frames with Alpha

MOV

AVAssetReader

HEVC with Alpha

Video Frames with Alpha
Decoding

Sample Buffers

VTDecompressionSession

Video Frames with Alpha
## APIs

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Playback</th>
<th>Decoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVAssetWriter</td>
<td>AVPlayer</td>
<td>AVAssetImageGenerator</td>
</tr>
<tr>
<td>AVAssetExportSession</td>
<td>AVPlayerItemVideoOutput</td>
<td>AVAssetReader</td>
</tr>
<tr>
<td>VTCompressionSession</td>
<td></td>
<td>VTDecompressionSession</td>
</tr>
</tbody>
</table>
For Example
import AVFoundation

let outputSettings = [AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
                      AVVideoWidthKey: width,
                      AVVideoHeightKey: height] as [String: Any]

let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
// AVAssetWriterInput - CodecType

import AVFoundation

let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height] as [String: Any]

let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
Premultiplied Alpha

Premultiplied Alpha

Straight Alpha
Premultiplied Alpha

Premultiplied Alpha

Straight Alpha

Both Premultiplied Alpha and Straight Alpha are supported
Both Premultiplied Alpha and Straight Alpha are supported

Premultiplied Alpha is recommended
import AVFoundation
import VideoToolbox

let outputSettings = {
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height
    kVTCompressionPropertyKey_AlphaChannelMode:
    kVTAphaChannelMode_PremultipliedAlpha
} as [String: Any]

let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)

CVBufferSetAttachment(
    pixelBuffer,
    CVImageBufferAlphaChannelModeKey,
    kCVImageBufferAlphaChannelMode_PremultipliedAlpha,
    .shouldPropagate)
// AVAssetWriterInput
import AVFoundation
import VideoToolbox

let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height
    kVTCompressionPropertyKey_AlphaChannelMode:
    kVTAlphaChannelMode_PremultipliedAlpha
] as [String: Any]

let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)

CVBufferSetAttachment(
    pixelBuffer,
    CVImageBufferAlphaChannelModeKey,
    kCVImageBufferAlphaChannelMode_PremultipliedAlpha,
    .shouldPropagate)
// AVAssetWriterInput
import AVFoundation
import VideoToolbox
let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height
    kVTCompressionPropertyKey_AlphaChannelMode:
    kVTAlphaChannelMode_PremultipliedAlpha
] as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)

CVBufferSetAttachment(
    pixelBuffer,
    CVImageBufferAlphaChannelModeKey,
    kCVImageBufferAlphaChannelMode_PremultipliedAlpha,
    .shouldPropagate)
Bitrate and Quality

Can control bitrate of base layer (bits per second)

Can control quality of alpha layer (0 to 1)
import AVFoundation
import VideoToolbox

let outputSettings = [
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,  
    AVVideoWidthKey: width,   
    AVVideoHeightKey: height,   
    AVVideoCompressionPropertiesKey: [
        kVTCompressionPropertyKey_AverageBitRate: baseLayerBitrate,   
        kVTCompressionPropertyKey_TargetQualityForAlpha: alphaLayerQuality
    ] as [String: Any]

let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
// AVAssetWriterInput - BitRate and Quality tradeoffs
import AVFoundation
import VideoToolbox
let outputSettings = {
    AVVideoCodecKey: AVVideoCodecType.hevcWithAlpha,
    AVVideoWidthKey: width,
    AVVideoHeightKey: height,
    AVVideoCompressionPropertiesKey: {
        kVTCompressionPropertyKey_AverageBitRate: baseLayerBitrate,
        kVTCompressionPropertyKey_TargetQualityForAlpha: alphaLayerQuality
    }
} as [String: Any]
let videoWriter = AVAssetWriterInput(mediaType: .video, outputSettings: outputSettings)
import AVFoundation

let hevcWithAlphaTracks = asset.tracks(withMediaCharacteristic:.containsAlphaChannel)

let containsAlphaChannel = formatDescription.extensions[.containsAlphaChannel] as! Bool?
// Presence of Alpha Channel

import AVFoundation

let hevcWithAlphaTracks = asset.tracks(withMediaCharacteristic:.containsAlphaChannel)

let containsAlphaChannel = formatDescription.extensions[.containsAlphaChannel] as! Bool?
// Presence of Alpha Channel

```swift
import AVFoundation

let hevcWithAlphaTracks = asset.tracks(withMediaCharacteristic:.containsAlphaChannel)

let containsAlphaChannel = formatDescription.extensions[.containsAlphaChannel] as! Bool?
```
import AVFoundation

AVAssetExportSession.determineCompatibility(
    ofExportPreset: AVAssetExportPresetHEVCHighestQualityWithAlpha,
    with: asset,
    outputFileType: .mov) { compatible in
        if compatible {
            // Export
        } else {
            print("Export Session failed compatibility check")
            // Handle failure
        }
    }
// AVAssetExportSession - compatibility

import AVFoundation

AVAssetExportSession.determineCompatibility(ofExportPreset: AVAssetExportPresetHEVCHighestQualityWithAlpha, with: asset, outputFileType: .mov) { compatible in
    if compatible {
        // Export
    } else {
        print("Export Session failed compatibility check")
        // Handle failure
    }
}
import AVFoundation

// Setup to use preferred color background
let prototypeInstruction = AVMutableVideoCompositionInstruction()
prototypeInstruction.backgroundColor = preferredColor // CGColor with ColorSpace

let videoComposition = AVMutableVideoComposition(propertiesOf: asset,
                                           prototypeInstruction: prototypeInstruction)

// Export
exportSession.outputURL = destinationURL
exportSession.outputFileType = .mov
exportSession.videoComposition = videoComposition
exportSession.exportAsynchronously {
  // Handle completion
}
// AVAssetExportSession - backwards compatibility

import AVFoundation

// Setup to use preferred color background
let prototypeInstruction = AVMutableVideoCompositionInstruction()
prototypeInstruction.backgroundColor = preferredColor // CGColor with ColorSpace
let videoComposition = AVMutableVideoComposition(propertiesOf: asset,
    prototypeInstruction: prototypeInstruction)

// Export
exportSession.outputURL = destinationURL
exportSession.outputFileType = .mov
exportSession.videoComposition = videoComposition
exportSession.exportAsynchronously {
    // Handle completion
}
Interoperability Profile

Single video track with ‘hvc1’ codec type

- Two layers
  - Base layer: main profile; 4:2:0 video-range; nuh_layer_id 0
  - Alpha layer: main profile; 4:2:0 full-range, neutral chroma channel; nuh_layer_id 1

- Base and alpha layers share one VPS
  - VPS extension required — indicates presence of alpha channel

- Base and alpha layers have distinct PPS and SPS with different IDs
- Base and alpha layers must have identical frame types and dependency structure

- SEI alpha_channel_info message in hvcC in video format description
  - Indicates premultiplied alpha or straight alpha
HEVC Video with Alpha

Advanced lossy compression technology

Supported in iOS 13, tvOS 13, macOS Catalina

Hardware accelerated on recent devices

Integrated with media APIs

Integrated in Safari
More Information

https://developer.apple.com/wwdc19/506