What’s New in Photos APIs

Session 505

Eric Hanson, Photos Platform Evangelist
Hasan Adil, Photos Engineer
Andreas Karlsson, Photos Engineer
Create great experiences
High-performance library access
Respecting user privacy
Image Picker Improvements
Easier access to photos from your iOS app

Hasan Adil, Photos Engineer
"Vacation Diary" Would Like to Access Your Photos
This will enable you to choose photos to include in your diary pages.

Don't Allow  OK
How was your day

Really excited to be here at WWDC!

Weather today

72°F
Obtaining Metadata

Created Date

Format of the photo

Metadata

// protocol implementation
public func imagePickerController(_ picker: UIImagePickerController,
didFinishPickingMediaWithInfo info: [String : Any]) {
    if let imageURL = info[UIImagePickerControllerImageURL] as? URL {
        print(imageURL)
    }
}
Obtaining Metadata

Created Date

Format of the photo

Metadata

// protocol implementation
public func imagePickerController(_ picker: UIImagePickerController,
didFinishPickingMediaWithInfo info: [String : Any]) {
    if let imageURL = info[UIImagePickerControllerImageURL] as? URL {
        print(imageURL)
    }
}
Accessing New Formats

```swift
var imageExportPreset: UIImagePickerControllerImportExportPreset { get set }

let imagePicker = UIImagePickerController()
imagePicker.imageExportPreset = .compatible
self.present(imagePicker, animated: true, completion: nil)

let imagePicker = UIImagePickerController()
imagePicker.imageExportPreset = .current
self.present(imagePicker, animated: true, completion: nil)
```
Accessing New Formats

```swift
var imageExportPreset: UIImagePickerControllerImportExportPreset { get set }

let imagePicker = UIImagePickerController()
imagePicker.imageExportPreset = .compatible
self.present(imagePicker, animated: true, completion: nil)

let imagePicker = UIImagePickerController()
imagePicker.imageExportPreset = .current
self.present(imagePicker, animated: true, completion: nil)
```
Accessing New Formats

```swift
var imageExportPreset: UIImagePickerControllerImportExportPreset { get set }

let imagePicker = UIImagePickerController()
imagePicker.imageExportPreset = .compatible
self.present(imagePicker, animated: true, completion: nil)

let imagePicker = UIImagePickerController()
imagePicker.imageExportPreset = .current
self.present(imagePicker, animated: true, completion: nil)
```
Accessing New Formats

```swift
var imageExportPreset: UIImagePickerControllerImportExportPreset { get set }

let imagePicker = UIImagePickerController()
imagePicker.imageExportPreset = .compatible
self.present(imagePicker, animated: true, completion: nil)

let imagePicker = UIImagePickerController()
imagePicker.imageExportPreset = .current
self.present(imagePicker, animated: true, completion: nil)
```
Getting Videos

```swift
var videoExportPreset: String { get set }

import AVFoundation

let imagePicker = UIImagePickerController()
imagePicker.videoExportPreset = AVAssetExportPresetHighestQuality
self.present(imagePicker, animated: true, completion: nil)

// See AVAssetExportSession
```
Getting Videos

```swift
var videoExportPreset: String { get set }

import AVFoundation

let imagePicker = UIImagePickerController()
imagePicker.videoExportPreset = AVAssetExportPresetHighestQuality
self.present(imagePicker, animated: true, completion: nil)

// See AVAssetExportSession
```
Getting Videos

```swift
var videoExportPreset: String { get set }

import AVFoundation

let imagePicker = UIImagePickerController()
imagePicker.videoExportPreset = AVAssetExportPresetHighestQuality
self.present(imagePicker, animated: true, completion: nil)

// See AVAssetExportSession
```
Getting Videos

```swift
var videoExportPreset: String { get set }

import AVFoundation

let imagePicker = UIImagePickerController()
imagePicker.videoExportPreset = AVAssetExportPresetHighestQuality
self.present(imagePicker, animated: true, completion: nil)

// See AVAssetExportSession
```

public func UISaveVideoAtPathToSavedPhotosAlbum(_ videoPath: String, _ completionTarget: Any?, _ completionSelector: Selector?, _ contextInfo: UnsafeMutableRawPointer?)
Saving Photos and Videos


public func UISaveVideoAtPathToSavedPhotosAlbum(_ videoPath: String, _ completionTarget: Any?, _ completionSelector: Selector?, _ contextInfo: UnsafeMutableRawPointer?)
Getting a PHAsset

Support for PhotoKit clients

Easy way to get the PHAsset object

```swift
public func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [String : Any]) {
    if let asset = info[UIImagePickerControllerPHAsset] as? PHAsset {
        print(asset)
    }
}
```
Getting a PHAsset

Support for PhotoKit clients

Easy way to get the PHAsset object

```swift
public func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [String : Any]) {
    if let asset = info[UIImagePickerControllerPHAsset] as? PHAsset {
        print(asset)
    }
}
```
enum PHAssetMediaType : Int {
    case unknown
    case image
    case video
    case audio
}

struct PHAssetMediaSubtype : OptionSet {
    static var photoPanorama
    static var photoHDR
    static var photoScreenshot
    static var photoLive
    static var photoDepthEffect
    static var videoStreamed
    static var videoHighFrameRate
    static var videoTimelapse
}
enum PHAssetMediaType : Int {
    case unknown
    case image
    case video
    case audio
}

struct PHAssetMediaSubtype : OptionSet {
    static var photoPanorama
    static var photoHDR
    static var photoScreenshot
    static var photoLive
    static var photoDepthEffect
    static var videoStreamed
    static var videoHighFrameRate
    static var videoTimelapse
}
class PHAsset : PHObject {
    var playbackStyle: PHAssetPlaybackStyle { get }
}

enum PHAssetPlaybackStyle : Int {
    case unsupported
    case image
    case imageAnimated
    case livePhoto
    case video
    case videoLooping
}
// PHAsset Playback Style

class PHAsset : PHObject {
    var playbackStyle: PHAssetPlaybackStyle { get }
}

enum PHAssetPlaybackStyle : Int {
    case unsupported
    case image
    case imageAnimated
    case livePhoto
    case video
    case videoLooping
}
class PHAsset : PHObject {
    var playbackStyle: PHAssetPlaybackStyle { get }
}

enum PHAssetPlaybackStyle : Int {
    case unsupported
    case image
    case imageAnimated
    case livePhoto
    case video
    case videoLooping
}
Animated Image

```swift
// PlaybackStyle.imageAnimated
imageManager.requestImageData(for: asset,
    options: options)
{(data, dataUTI, orientation, info) in
    // Using AnimatedImageView from Sample App
    let animatedImage = AnimatedImage(data: data)
    animatedImageView.animatedImage = animatedImage
}
```
Animated Image

// PlaybackStyle.imageAnimated
imageManager.requestImageData(for: asset,
    options: options)
{(data, dataUTI, orientation, info) in
    // Using AnimatedImageView from Sample App
    let animatedImage = AnimatedImage(data: data)
    animatedImageView.animatedImage = animatedImage
}
Live Photo

```swift
// PlaybackStyle.livePhoto
imageManager.requestLivePhoto(for: asset,
    targetSize: pixelSize,
    contentMode: .aspectFill,
    options: options)

{(livePhoto, info) in
    // Using PHLivePhotoView
    livePhotoView.livePhoto = livePhoto
}
```
// PlaybackStyle.livePhoto
imageManager.requestLivePhoto(for: asset,
    targetSize: pixelSize,
    contentMode: .aspectFill,
    options: options)

{(livePhoto, info) in
    // Using PHLivePhotoView
    livePhotoView.livePhoto = livePhoto
}
Looping Video

// PlaybackStyle.videoLooping
imageManager.requestPlayerItem(forVideo: asset,
    options: options)
{
    playerItem, info in
        DispatchQueue.main.async {
            let player = AVQueuePlayer()
            playerLooper = AVPlayerLooper(player: player,
                templateItem: playerItem)
            playerLayer.player = player
            player.play()
        }
}
// PlaybackStyle.videoLooping
imageManager.requestPlayerItem(forVideo: asset,
  options: options)

{ playerItem, info in
  DispatchQueue.main.async {
    let player = AVQueuePlayer()
    playerLooper = AVPlayerLooper(player: player,
      templateItem: playerItem)
    playerLayer.player = player
    player.play()
  }
}

Test your app with a large photo library
Large Photo Libraries for Testing

New Sample App—Photo Library Filler

Download and run on oldest device

Generate 100,000 photos

100000
Add Photos
let assets = PHAsset.fetchAssets(with: options)
let assets = PHAsset.fetchAssets(with: options)
let assets = PHAsset.fetchAssets(with: options)
let assets = PHAsset.fetchAssets(with: options)
let options = PHFetchOptions()

options.predicate = NSPredicate(format: "isFavorite = %d", true)

options.sortDescriptors = [NSSortDescriptor(key: "creationDate", ascending: true)]

let assets = PHAsset.fetchAssets(with: options)
let options = PHFetchOptions()

options.predicate = NSPredicate(format: "isFavorite = %d", true)

options.sortDescriptors = [NSSortDescriptor(key: "creationDate", ascending: true)]

let assets = PHAsset.fetchAssets(with: options)
let options = PHFetchOptions()
options.predicate = NSPredicate(format: "isFavorite = %d", true)
options.sortDescriptors = [NSSortDescriptor(key: "creationDate", ascending: true)]
let assets = PHAsset.fetchAssets(with: options)
let options = PHFetchOptions()

options.predicate = NSPredicate(format: "isFavorite = %d", true)

options.sortDescriptors = [NSSortDescriptor(key: "creationDate", ascending: true)]

let assets = PHAsset.fetchAssets(with: options)
let smartAlbums = PHAssetCollection.fetchAssetCollections(with: .smartAlbum,
subetype: .smartAlbumFavorites,
options: nil)
let assets = PHAsset.fetchAssets(in: smartAlbums.firstObject, options: nil)
let options = PHFetchOptions()

options.predicate = NSPredicate(format: "isFavorite = %d", true)

options.sortDescriptors = [NSSortDescriptor(key: "creationDate", ascending: true)]

let assets = PHAsset.fetchAssets(with: options)

let smartAlbums = PHAssetCollection.fetchAssetCollections(with: .smartAlbum,
                                           subtype: .smartAlbumFavorites,
                                           options: nil)

let assets = PHAsset.fetchAssets(in: smartAlbums.firstObject, options: nil)
let assets = PHAsset.fetchAssets(with: options)
let assets = PHAsset.fetchAssets(with: options)
let assets = PHAsset.fetchAssets(with: options)
let assets = PHAsset.fetchAssets(with: options)
assets.enumerateObjects { (asset, index, stop) in
    // do something with the asset
}
let assets = PHAsset.fetchAssets(with: options)

assets.enumerateObjects { (asset, index, stop) in
    // do something with the asset
}
let assets = PHAsset.fetchAssets(with: options)
assets.enumerateObjects { (asset, index, stop) in
    // do something with the asset
}
let assets = PHAsset.fetchAssets(with: options)

assets.enumerateObjects { (asset, index, stop) in
    // do something with the asset
}
let assets = PHAsset.fetchAssets(with: options)
assets.enumerateObjects { (asset, index, stop) in
    // do something with the asset
}
```swift
let assets = PHAsset.fetchAssets(with: options)

assets.enumerateObjects { (asset, index, stop) in
    // do something with the asset
}
```

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifiers</td>
<td>34</td>
<td>235</td>
<td>65</td>
<td>32</td>
<td>87</td>
<td>75</td>
<td>231</td>
<td>39</td>
<td>54</td>
</tr>
<tr>
<td>Objects</td>
<td>PHAsset</td>
<td>PHAsset</td>
<td>PHAsset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
let assets = PHAsset.fetchAssets(with: options)

assets.enumerateObjects { (asset, index, stop) in
    // do something with the asset
}
let assets = PHAsset.fetchAssets(with: options)

assets.enumerateObjects { (asset, index, stop) in
    // do something with the asset
}
10s
to enumerate a large fetch result
let someAsset = // Asset 75
let assets = PHAsset.fetchAssets(with: options)
let index = assets.indexOfObject(someAsset)
let contains = assets.containsObject(someAsset)
Finding Assets in a PHFetchResult

```swift
let someAsset = // Asset 75
let assets = PHAsset.fetchAssets(with: options)
let index = assets.indexOfObject(someAsset)
let contains = assets.containsObject(someAsset)
```

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifiers</td>
<td>34</td>
<td>235</td>
<td>65</td>
<td>32</td>
<td>87</td>
<td>75</td>
<td>231</td>
<td>39</td>
<td>54</td>
</tr>
</tbody>
</table>

Objects
Finding Assets in a PHFetchResult

```swift
let someAsset = // Asset 75
let assets = PHAsset.fetchAssets(with: options)
let index = assets.indexOfObject(someAsset)
let contains = assets.containsObject(someAsset)
```
Finding Assets in a PHFetchResult

```swift
let someAsset = // Asset 75
let assets = PHAsset.fetchAssets(with: options)
let index = assets.indexOfObject(someAsset)
let contains = assets.containsObject(someAsset)
```
Finding Assets in a PHFetchResult

```swift
let someAsset = // Asset 75
let assets = PHAsset.fetchAssets(with: options)
let index = assets.indexOfObject(someAsset)
let contains = assets.containsObject(someAsset)
```
Finding Assets in a PHFetchResult

```
let someAsset = // Asset 75
let assets = PHAsset.fetchAssets(with: options)
let index = assets.indexOfObject(someAsset)
let contains = assets.containsObject(someAsset)
```
Test your app with a large photo library
Photos Project Extensions
Bring your app to macOS Photos

Eric Hanson, Photos Platform Evangelist
Dog Show
May 20, 2017
First Photos
Demo
One more. In action.
How it all works
NSExtension Point

Similar to Photos editing extensions
NSExtension Point

Similar to Photos editing extensions

New extension point: com.apple.photo-project
NSExtension Point

Similar to Photos editing extensions

New extension point: com.apple.photo-project

New Xcode template
NSExtension Point

Similar to Photos editing extensions

New extension point: com.apple.photo-project

New Xcode template

Automatic discovery of your app inside Photos
NSExtension Point

Similar to Photos editing extensions

New extension point: com.apple.photo-project

New Xcode template

Automatic discovery of your app inside Photos

Mac App Store link to find extension apps
PHProjectExtensionController
Principal controller in your extension

Photos App
Main View Controller
PhotoKit

Your App
Extension
Extension Code
Shared Assets and Code
App Code
PHProjectExtensionController
Principal controller in your extension

Photos App
Main View Controller
PhotoKit

Your App
Extension
Extension View Controller
Extension Code
Shared Assets and Code
App Code
PHProjectExtensionController
Principal controller in your extension

- PHProjectExtentionController
- PHProjectExtensionContext
- PHProjectInfo
- PhotoKit
- Main View Controller
- Photos App

Extension View Controller
Extension Code
Shared Assets and Code
App Code
Your App
PHProjectExtensionController
Principal controller in your extension

- PHProjectExtensionContext
- PHProjectInfo
- Extension View Controller
- Extension Code
- Shared Assets and Code
- App Code

Main View Controller
- Extension View Controller
- PhotoKit

Photos App
Your App
public protocol PHProjectExtensionController : NSObjectProtocol {

    optional public var supportedProjectTypes: [PHProjectTypeDescription] { get }

    // Called the first time a project is created
    public func beginProject(with extensionContext: PHProjectExtensionContext,
                            projectInfo: PHProjectInfo, completion: @escaping (Error?) -> Void)

    // Called anytime the user returns to a project that was previously created
    public func resumeProject(with extensionContext: PHProjectExtensionContext,
                               completion: @escaping (Error?) -> Void)

    // Called when a user is switching away from the project
    public func finishProject(completionHandler completion: @escaping () -> Void)
}

public protocol PHProjectExtensionController : NSObjectProtocol {

optional public var supportedProjectTypes: [PHProjectTypeDescription] { get }

// Called the first time a project is created
public func beginProject(with extensionContext: PHProjectExtensionContext,
                          projectInfo: PHProjectInfo, completion: @escaping (Error?) -> Void)

// Called anytime the user returns to a project that was previously created
public func resumeProject(with extensionContext: PHProjectExtensionContext,
                          completion: @escaping (Error?) -> Void)

// Called when a user is switching away from the project
public func finishProject(completionHandler completion: @escaping () -> Void)
}
public protocol PHProjectExtensionController : NSObjectProtocol {

    optional public var supportedProjectTypes: [PHProjectTypeDescription] { get }

    // Called the first time a project is created
    public func beginProject(with extensionContext: PHProjectExtensionContext,
                              projectInfo: PHProjectInfo, completion: @escaping (Error?) -> Void)

    // Called anytime the user returns to a project that was previously created
    public func resumeProject(with extensionContext: PHProjectExtensionContext,
                              completion: @escaping (Error?) -> Void)

    // Called when a user is switching away from the project
    public func finishProject(completionHandler completion: @escaping () -> Void)
}
public protocol PHProjectExtensionController : NSObjectProtocol {

    optional public var supportedProjectTypes: [PHProjectTypeDescription] { get }

    // Called the first time a project is created
    public func beginProject(with extensionContext: PHProjectExtensionContext,
                              projectInfo: PHProjectInfo, completion: @escaping (Error?) -> Void)

    // Called anytime the user returns to a project that was previously created
    public func resumeProject(with extensionContext: PHProjectExtensionContext,
                               completion: @escaping (Error?) -> Void)

    // Called when a user is switching away from the project
    public func finishProject(completionHandler completion: @escaping () -> Void)
}
public protocol PHProjectExtensionController : NSObjectProtocol {

    optional public var supportedProjectTypes: [PHProjectTypeDescription] { get }

// Called the first time a project is created
public func beginProject(with extensionContext: PHProjectExtensionContext,
    projectInfo: PHProjectInfo, completion: @escaping (Error?) -> Void)

// Called anytime the user returns to a project that was previously created
public func resumeProject(with extensionContext: PHProjectExtensionContext,
    completion: @escaping (Error?) -> Void)

// Called when a user is switching away from the project
public func finishProject(completionHandler completion: @escaping () -> Void)
}
PHProjectExtensionContext

Your access to the project and photo library

PHProject

PHPhotoLibrary
PHProject

Intended for project state

Layout and asset references, not image data

Limited to 1 MB

```swift
// PHProject.h
// Photos

class PHProject : PHAssetCollection {
    var projectExtensionData: Data { get }
}
```
PHProject

Intended for project state

Layout and asset references, not image data

Limited to 1 MB

// PHProject.h
// Photos

class PHProject : PHAssetCollection {
    var projectExtensionData: Data { get }
}

Title and key asset, and data

Extension’s project data

```swift
do {
    let changeRequest = PHProjectChangeRequest(project: self.project)

    try self.library.performChangesAndWait {
        changeRequest.projectExtensionData = 
            NSKeyedArchiver.archivedData(withRootObject: cloudIdentifiers)
    }
}

} catch {
    print("Failed to save project data: \(error.localizedDescription)")
}
do {
    let changeRequest = PHProjectChangeRequest(project: self.project)

    try self.library.performChangesAndWait {
        changeRequest.projectExtensionData = 
            NSKeyedArchiver.archivedData(withRootObject: cloudIdentifiers)
    }
}

} catch {
    print("Failed to save project data: \(error.localizedDescription)")
}
do {
    let changeRequest = PHProjectChangeRequest(project: self.project)

    try self.library.performChangesAndWait {
        changeRequest.projectExtensionData = 
            NSKeyedArchiver.archivedData(withRootObject: cloudIdentifiers)
    }
}

} catch {
    print("Failed to save project data: \(error.localizedDescription)"
}
Where the magic begins
AT THE BEACH
2013 - 2016

Photos
PHProjectInfo
All the context you need at project creation
// PHProjectInfo
var sections: [PHProjectSection]
PHProjectInfo
All the context you need at project creation

// PHProjectInfo
var sections: [PHProjectSection]

// PHProjectSection
var sectionType: PHProjectSection.SectionType
var sectionContents: [PHProjectSectionContent]
PHProjectInfo
All the context you need at project creation

// PHProjectInfo
var sections: [PHProjectSection]

// PHProjectSection
var sectionType: PHProjectSection.SectionType
var sectionContents: [PHProjectSectionContent]

// PHProjectSectionContent
var elements: [PHProjectElement]
var numberOfColumns: Int
var aspectRatio: Double
var cloudAssetIdentifiers: [PHCloudIdentifier]
Multiple content objects represent different curation levels

<table>
<thead>
<tr>
<th>Level</th>
<th>Section Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image1.png" alt="Images" /></td>
</tr>
<tr>
<td>1</td>
<td><img src="image2.png" alt="Images" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image3.png" alt="Images" /></td>
</tr>
</tbody>
</table>
class PHProjectSectionContent: NSObject, NSSecureCoding {
    var elements: [PHProjectElement] { get }
    var numberOfColumns: Int { get }
    var aspectRatio: Double { get }
    var cloudAssetIdentifiers: [PHCloudIdentifier] { get }
}
class PHProjectSectionContent : NSObject, NSSecureCoding {
    var elements: [PHProjectElement] { get }
    var numberOfColumns: Int { get }
    var aspectRatio: Double { get }
    var cloudAssetIdentifiers: [PHCloudIdentifier] { get }
}
Asset identifiers in PHProjectInfo are PHCloudIdentifiers

Convert to local identifiers to fetch PHAssets

Always persist cloud identifiers in saved project data

```swift
// get content cloud identifiers
cloudIdentifiers += dataDict.value(forKey: "contentIdentifiers") as! [PHCloudIdentifier]

// convert to local identifiers
let localIdentifiers = self.library.localIdentifiers(for: cloudIdentifiers)
```
Asset identifiers in PHProjectInfo are PHCloudIdentifiers

Convert to local identifiers to fetch PHAssets

Always persist cloud identifiers in saved project data

```swift
// get content cloud identifiers
cloudIdentifiers += dataDict.value(forKey: "contentIdentifiers") as! [PHCloudIdentifier]

// convert to local identifiers
let localIdentifiers = self.library.localIdentifiers(for: cloudIdentifiers)
```
class PHProjectSectionContent : NSObject, NSSecureCoding {
    var elements: [PHProjectElement] { get }
    var numberOfColumns: Int { get }
    var aspectRatio: Double { get }
    var cloudAssetIdentifiers: [PHCloudIdentifier] { get }
}
// PHProjectSectionContent

class PHProjectSectionContent : NSObject, NSSecureCoding {
    var elements: [PHProjectElement] { get }
    var numberOfColumns: Int { get }
    var aspectRatio: Double { get }
    var cloudAssetIdentifiers: [PHCloudIdentifier] { get }
}
Grid Coordinates
Grid Coordinates
Grid Coordinates

numberOfColumns = 20
Grid Coordinates
placement = (0,0,8,9)
// PHProjectSectionContent

class PHProjectSectionContent : NSObject, NSSecureCoding {
    var elements: [PHProjectElement] { get }
    var numberOfColumns: Int { get }
    var aspectRatio: Double { get }
    var cloudAssetIdentifiers: [PHCloudIdentifier] { get }
}
class PHProjectSectionContent : NSObject, NSSecureCoding {
    var elements: [PHProjectElement] { get }
    var numberOfColumns: Int { get }
    var aspectRatio: Double { get }
    var cloudAssetIdentifiers: [PHCloudIdentifier] { get }
}
// PHProjectElement

class PHProjectElement : NSObject, NSSecureCoding {

    // Relative significance in range from 0.0 to 1.0. Default is 0.5.
    var weight: Double { get }

    // Positioning in grid space coordinates for element if honoring layout grid
    var placement: CGRect { get }
}

class PHProjectElement : NSObject, NSSecureCoding {

    // Relative significance in range from 0.0 to 1.0. Default is 0.5.
    var weight: Double { get }

    // Positioning in grid space coordinates for element if honoring layout grid
    var placement: CGRect { get }
}
// PHProjectElement

class PHProjectElement : NSObject, NSSecureCoding {

    // Relative significance in range from 0.0 to 1.0. Default is 0.5.
    var weight: Double { get }

    // Positioning in grid space coordinates for element if honoring layout grid
    var placement: CGRect { get }

}
class PHProjectAssetElement : PHProjectElement, NSSecureCoding {

    // convert to local identifier to fetch PHAsset
    var cloudAssetIdentifier: PHCloudIdentifier { get }
    var annotation: String { get }
    var cropRect: CGRect { get }
    var regionsOfInterest: [PHProjectRegionOfInterest] { get }
}
class PHProjectAssetElement : PHProjectElement, NSSecureCoding {

    // convert to local identifier to fetch PHAsset
    var cloudAssetIdentifier: PHCloudIdentifier { get }
    var annotation: String { get }
    var cropRect: CGRect { get }
    var regionsOfInterest: [PHProjectRegionOfInterest] { get }
}
Demo
Let’s build our own
Summary

Take advantage of frictionless photo picker

Use new media presentation types

Test with large libraries

Create photo services with Photos Project Extensions
More Information

<table>
<thead>
<tr>
<th>Session Description</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vision Framework: Building on Core ML</td>
<td>Hall 2</td>
<td>Wednesday 3:10PM</td>
</tr>
<tr>
<td>Capturing Depth in iPhone Photography</td>
<td>Executive Ballroom</td>
<td>Wednesday 5:10PM</td>
</tr>
<tr>
<td>Photography Get Together</td>
<td>Technology Lab J</td>
<td>Wednesday 6:30PM</td>
</tr>
<tr>
<td>Imaging Editing with Depth</td>
<td>Grand Ballroom A</td>
<td>Thursday 11:00AM</td>
</tr>
<tr>
<td>Advances in Core Image: Filters, Metal, Vision, and More</td>
<td>Executive Ballroom</td>
<td>Thursday 1:50PM</td>
</tr>
<tr>
<td>Working with HEIF and HEVC</td>
<td>Hall 2</td>
<td>Friday 11:00AM</td>
</tr>
</tbody>
</table>
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vision Framework: Building on Core ML</td>
<td>Hall 2</td>
<td>Wednesday 3:10PM</td>
</tr>
<tr>
<td>Capturing Depth in iPhone Photography</td>
<td>Executive Ballroom</td>
<td>Wednesday 5:10PM</td>
</tr>
<tr>
<td>Photography Get Together</td>
<td>Technology Lab J</td>
<td>Wednesday 6:30PM</td>
</tr>
<tr>
<td>Imaging Editing with Depth</td>
<td>Grand Ballroom A</td>
<td>Thursday 11:00AM</td>
</tr>
<tr>
<td>Advances in Core Image: Filters, Metal, Vision, and More</td>
<td>Executive Ballroom</td>
<td>Thursday 1:50PM</td>
</tr>
<tr>
<td>Working with HEIF and HEVC</td>
<td>Hall 2</td>
<td>Friday 11:00AM</td>
</tr>
<tr>
<td>Lab Name</td>
<td>Location</td>
<td>Days</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>--------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Photos Extensions and PhotoKit Lab</td>
<td>Technology Lab A</td>
<td>Thur</td>
</tr>
<tr>
<td>Photos Editing and Core Image Lab</td>
<td>Technology Lab F</td>
<td>Thur</td>
</tr>
<tr>
<td>Photos Depth and Capture Lab</td>
<td>Technology Lab A</td>
<td>Thur</td>
</tr>
<tr>
<td>HEIF and HEVC Lab</td>
<td>Technology Lab F</td>
<td>Fri</td>
</tr>
<tr>
<td>Vision Lab</td>
<td>Technology Lab A</td>
<td>Fri</td>
</tr>
<tr>
<td>Photos Depth and Capture Lab</td>
<td>Technology Lab F</td>
<td>Fri</td>
</tr>
</tbody>
</table>