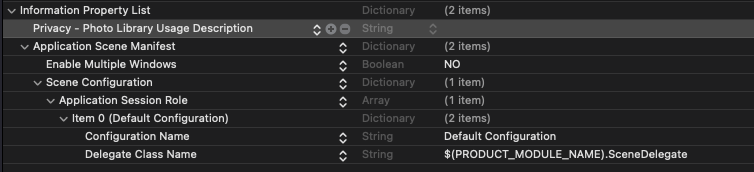
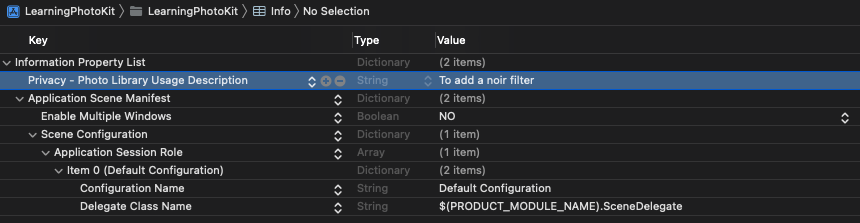
# PhotoKit

Các App Photos quản lý các tài nguyên của ảnh thông qua tập hợp các APIs gọi là PhotoKit.

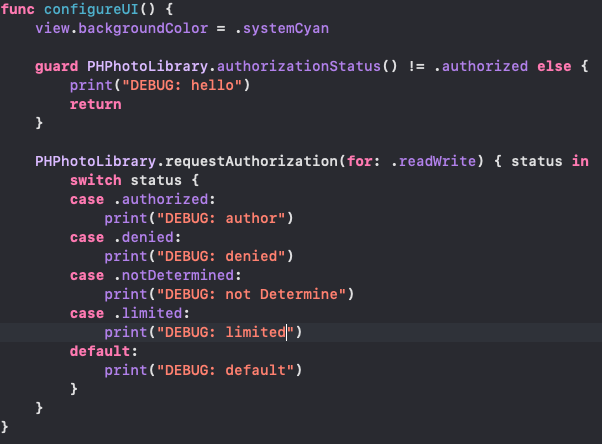
## Getting PhotoKit Permissions

Giống như mọi APIs IOS, PhotoKit sử dụng permissions model. Nó sẽ hiển thị ra 1 dialog yêu cầu các quyền để có thể truy cập vào ảnh của người dụng. Chúng ta sử dụng **PHPhotoLibrary** để yêu cầu quyền truy cập **photo library**.

* Để có thể yêu cầu quyền truy cập ảnh từ thư viện, đầu tiên ta phải chỉnh sửa **info.plist**
* Mở Info.plist
* Right-click **Information Property List** and select Add Row. A new line appears.
* Nhập Key **NSPhotoLibraryUsageDescription** and press enter.
* Ở cột value cùng hàng, ta có thể nhập bất kì value nào hoặc ko nhập gì cũng được, nếu nhập **To add a noir filter** thì khi yêu cầu sẽ hiển thị mô ta như ở ảnh trên, nếu ko nhập gì sẽ mô tả như ảnh dưới

* Để có thể yêu cầu quyền truy cập, ta **import Photos**, và sử dụng đoạn code sau:



* Với **PHPhotoLibrary.authorizationStatus()** để kiểm tra xem người dùng đã cho phép truy cập vào thư viện hay chưa, nếu đã cho phếp thì print “hello”. **Lưu ý rằng cho phép “Tất cả ảnh” hoặc “Chọn ảnh…” đều được tính là đã cho phép truy cập và sẽ in hello**. Nếu chưa cho phép thì ta yêu cầu quyền truy cập thông quá **PHP.requestAuthorization()**, lưu ý là hàm này chỉ được gọi 1 lần khi mới cài đặt app và chạy app lần đầu tiên.

## Understanding Assests

Mặc dù mục đích chính là nhận được các images, nhưng 1 điều quan trọng là ta lại hầu hết làm việc với assets trong PhotoKit. Các thông tin như favorites, vị trí của ảnh đều là metadata. Apple cung cấp cho ta **PHAsset** để quản lý các thông tin metadata của 1 ảnh. **PHAsset** có thể thay đổi được, và không chứa ảnh, mà chỉ chứa các thông tin liên quan đến ảnh. Các thông tin như là thời gian tạo, thời gian sửa đổi, favorite, trạng thái hiddens,…

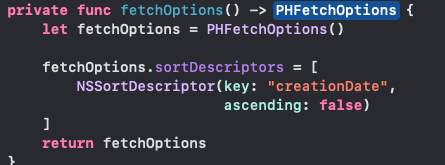
Đôi khi ta làm việc với **PHAssetCollection**. Đây là 1 nhóm các assets.

## Các bước load ảnh từ Assest

* Ta sẽ fetchAssests thông qua lệnh như ở dưới đây:

No description available.

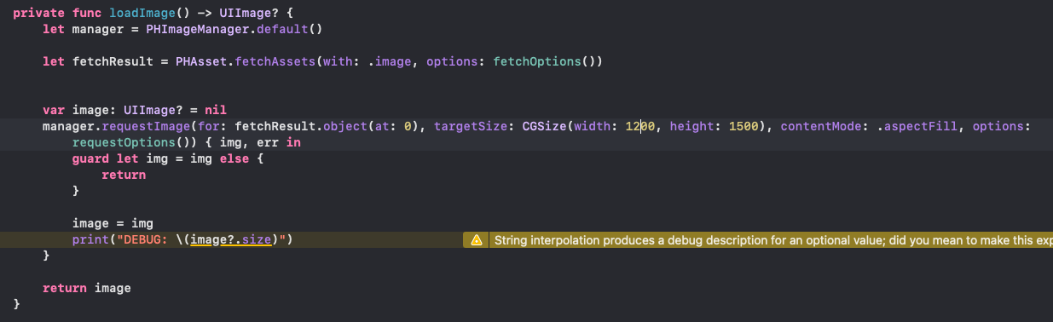
Hàm này sẽ trả về tất cả các Assests của tất cả ảnh được sắp xếp theo thứ tự nào đó, Các tham số truyền vào là **.image(fetch asset của imgae)** và **options(**Đây chính là thứ quy đinh thứ tự của Assest**).**

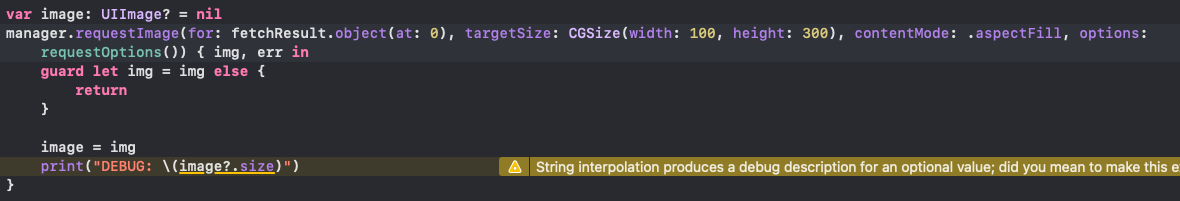


Thứ tự sắp xếp của Assest sẽ theo key là “creationDate”. Đây là thời gian mà ảnh được tạo chứ ko phải thời gian lưu về máy. Ví dụ ta chụp ảnh bằng điện thoại của mình, thì thời điểm mà ta chụp chính là thới điểm ảnh được tạo(“**creationDate**”). Khi ta chuyển ảnh đó lên mạng và tải lại về máy, đó ko phải “creationDate”. **Ascending là tăng dần. Các từ khóa tương tự là:**

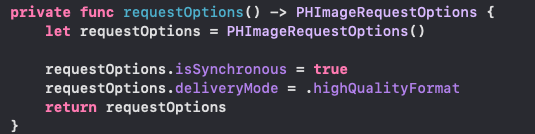
* **modificationDate:** The date and time when the asset was last modified (including edits and metadata changes).
* **duration**: The duration of the asset (for example, the length of a video or a Live Photo).0
* **favorite**: Whether the asset has been marked as a favorite by the user (as a boolean value). Cái này sẽ sắp xếp mấy thằng favorite lên trước, mấy thằng ko có favorite ở sau
* **mediaType**: The media type of the asset, such as image, video, or audio (as an integer value, where 1 represents image, 2 represents video, and 3 represents audio).
* **pixelWidth**: The width of the asset's image or video, in pixels.
* **pixelHeight**: The height of the asset's image or video, in pixels.
* **localizedTitle**: The title of the asset, as displayed in the Photos app (as a string value).
* Kết quả nhận được là **fetchResult, sẽ chứa tập hợp các Assets của ảnh trong library.** Ta truy cập các **Assest đó** thông qua **fetchResult.object(at: )**. Và để lấy được Image ta sử dụng đoạn Code sau:

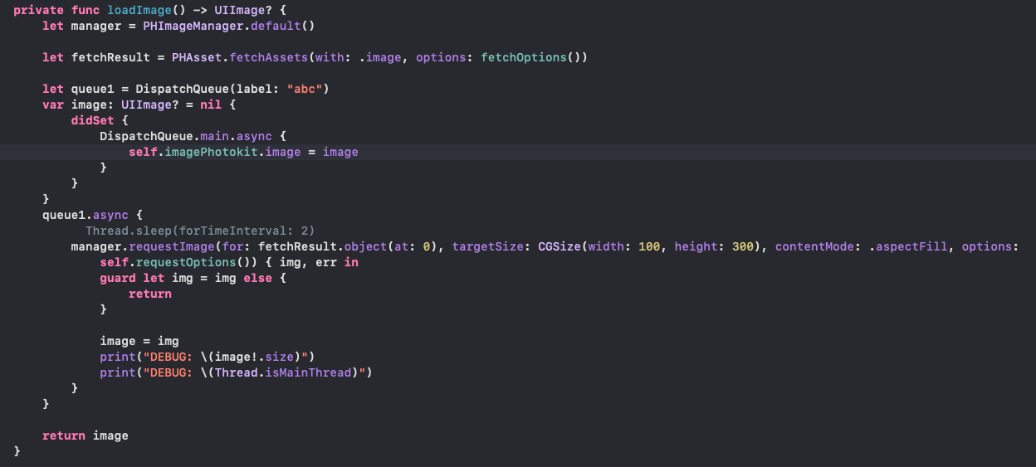


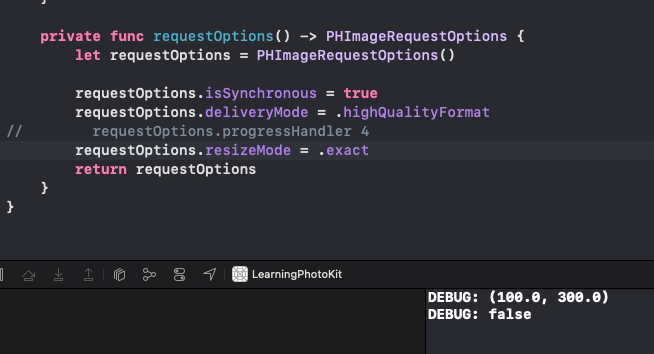
* **fetchResult.object(at: 0)**: Lấy Assest đầu tiên
* No description available.**targetSize**: Đây là size ảnh ta mong muốn nhận được. Size thực tế nhận được sẽ gần với targetSize nhất có thể trong **khi vẫn duy trì được tỉ lệ kích thước của ảnh gốc**(If the original image has a different aspect ratio than the target size, the returned image will be cropped or padded as necessary to fit the target size.)

No description available.**Ta thấy rằng tỉ lệ 630/373 gần như bằng 300/178. 🡪 Chứng tỏ vẫn giữ tỉ lệ ảnh gốc**

* No description available.No description available.
* Đối với requestOption() ta có thư viện **PHImageRequestOptions(), thư viện này cung cấp thêm các options để lấy images.** Thư viện này cung cấp các thuộc tính sau:



* + No description available.**isSynchoronous:** thuộc tính này chỉ định **request** sẽ được thực hiện synchronously or asynchronously. Nếu là **true, request** sẽ block thread hiện tại cho đến khi lấy được ảnh chất lượng cao nhất có thể. Nếu là **false,** request sẽ lập tức return khi lấy được ảnh theo định dạng , và sẽ lập tức chạy closure **resultHandler**(là 1 parameter của **requestImage**) là **false**, sẽ không lấy được image, nên muốn lấy ảnh ta
* No description available.
  + **resizeMode**: An enum value that specifies the desired resizing mode for the image. The possible values are .none, .fast, and .exact. **If you want to request an image at a specific size, you can set resizeMode to .exact, and then set the targetSize property to the desired size.** Ta thấy ảnh dưới có size đúng là (100, 300) như ta xét bên trên chứ ko theo tỉ lệ nữa.

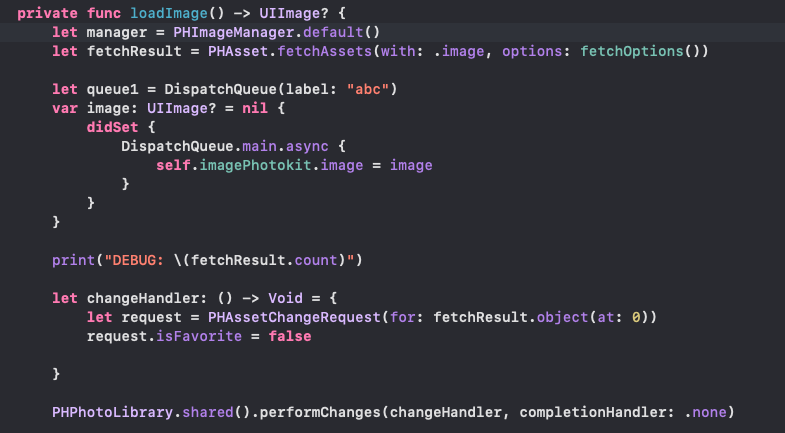


* + **deliveryMode:** Chỉ định chất lượng cho hình ảnh. Nếu **là .opportunistic,** hệ thống sẽ trả về hình ảnh cao nhất có thể, nhưng có thể trả về chất lượng thấp hơn nếu chất lượng thấp hơn có sẵn. Nếu là **.highQualityFormat**, hệ thống sẽ luôn trả về chất lượng tốt nhất, kể cả có tốn nhiều thời gian. Với **.fastFormat** hệ thống sẽ trả về chất lượng thấp nhất để lấy **image** nhanh nhất
  + **isNetworkAccessAllowed**: Aboolean value that indicates whether network access is allowed to retrieve the image. If true, the system may use the network to retrieve the image if it is not available locally. If false, the system will only retrieve the image if it is available locally.
  + Ta cũng có thể sử dụng **predicate** để lọc bớt ảnh, như chỉ lấy các ảnh đã được xét là **favorite như sau:**



## Chỉnh sửa metadata

Chúng ta chỉnh sửa metadata thông qua đoạn code sau đây:

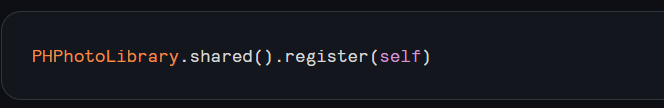


Đoạn **performChanges** đó 😊) Ảnh đầu tiền được đánh dấu là **favorite** sẽ được đánh dấu là ko **favorite** nữa 😊)).

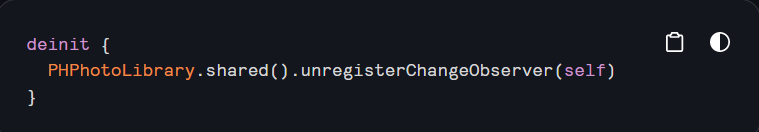
## Photo View Controller Change Observe

PhotoKit **caches** the results of fetch requests for better performance. When you tap the favorite button**, the asset updates in the library, but the view controller’s copy of the asset is now out of date**. The controller needs **to listen for updates** to the library and update its asset when necessary. Do this by conforming the controller to **PHPhotoLibraryChangeObserver.**

* Để nhận được updates từ library, ta sử cần register với lệnh sau:

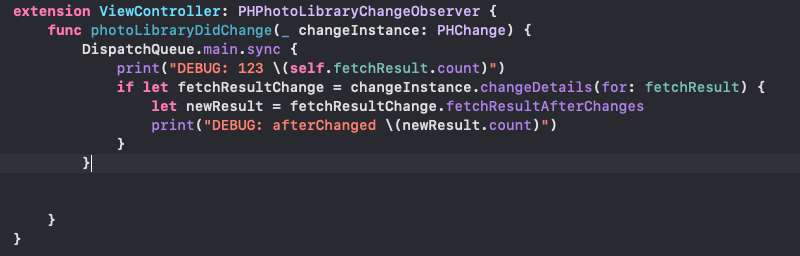


* Và để tránh leak memory, ViewController cần unregister khi ko cần thiết

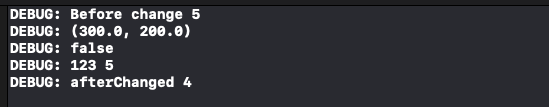


Chú ý là:

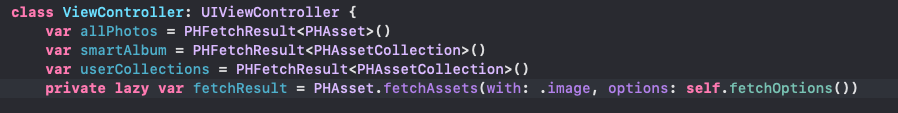
* Để nhận được updates ta kế thừa Protocol **PHPhotoLibraryChangeObserver.**



* + Đoạn này có DispatchQueue hay ko cũng được nhé, nhưng mà khả năng sẽ reload table hoặc collectionView ở đây nên thêm cũng được.
  + **changeInstance** sẽ nắm những thông tin thay đổi của 1 **fetchResult** hoặc của 1 **Asset.** Nó cung cấp hàm đa hình changeDetail() với tham số truyền vào khác nhau có thể là **1 Asset cụ thể hoặc 1 fetchResult cụ thể.** Thường thị người dùng sẽ thay đổi nhiều ảnh 1 lúc nên hay dùng **fetchResult.** Hàm này sẽ hoạt động bằng cách truyền 1 **Asset hoặc 1 fetchResult vào.**  Nó sẽ compare các thông tin hiện tại của **Asset hoặc fetchResult** với các thông tin trong **changeInstance.** Nếu nó thấy khác biệt sẽ trả lại 1 biến mới **fetchResultChange.** Và ta cập nhập thông tin mới qua **fetchResultChange.fetchResultAfterChanges. Sau khi dislike 1 ảnh đi thì số ảnh favorite giảm còn 4**



* + Có 1 điểm lưu ý là ở View Controller, vừa nãy khai báo biến **fetchResult** thuộc kiểu lazy var, và khi thay đổi **favorite hay delete ảnh đi thì** không thay đổi gì. Bởi vì **fetchResult là biến lazy var và** sẽ được gọi lần đầu trong **changeInstance.changeDetail(),** dẫn đến tại thời điểm đấy thì nó mới được gán giá trị. Mà lúc đó là thời điểm sau khi **user thay đổi thông tin ảnh.** Dẫn đến **changeInstance** so sánh thông tin **fetchResult** và trả về nil(Không có thông tin được thay đổi). Để tránh lỗi ngu học này thì ta phải gọi thằng **fetchResult trước khi gọi nó trong hàm photoLibrariDidChange.**



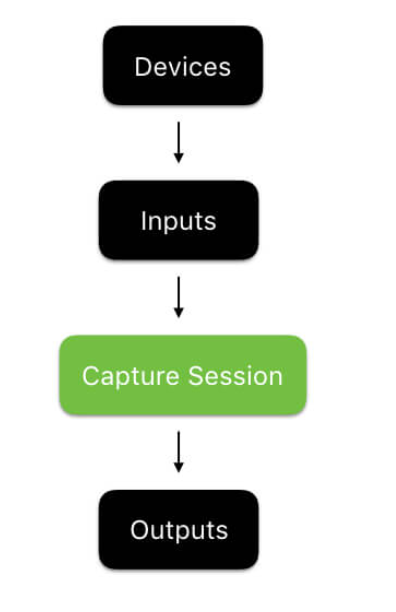
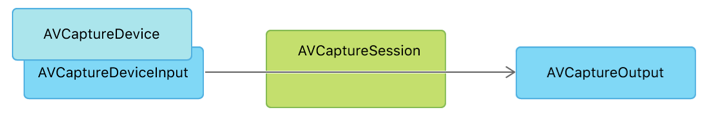
# AVFoundation

AVFoundation là 1 frameword cho phép ta làm việc với capture, xử lý, chỉnh sửa ảnh, audio và video trên các thiết bị Apple. Trong phần này , chúng ta sẽ thực hiện các chức năng như chụp ảnh, video với các camera trước, sau, flash,…

## 2.1 Core components of capuring photos và video

Trong AVFoundation, thành phần core của việc chụp ảnh, videos là **capture session.** Theo như Apple, **capture session** là 1 đối tượng quản lý các hành động capture và điều phối luồng dữ liệu từ input thiết bị tới capture outputs. Trong **AVFoundation**, **capture session** được quản lý bởi **.**

Ta cũng có, **capture device được quản lý bởi AVCaptureDevice,** đây là 1 phần mềmđược sử dụng để truy cập vào các thiết bị vật lý mà được sử dụng để chụp ảnh và video. Luồng dữ liệu cơ bản là, ta sử dụng **capture device,** sử dụng nó để tạo ra **capture inputs**, sau đó sẽ cung cấp input đó **capture session,** cuối cùng là lưu kết quả vào **capture outputs.**



## **2.2 Các bước thực thi**

### 2.2.1 Setting AVCaptureSession

Các nhiệm vụ chính của 1 AVCaptureSession:

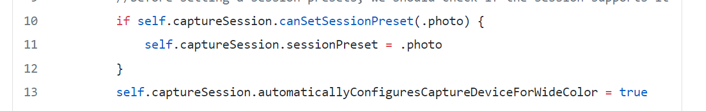
* Manage Inputs và Outputs
* Manage Running State
* Manage Connections
* Manage Color Spaces

Các bước Setting:

* Import AVFoundation
* Tạo 1 session, và sử dụng phương thức **beginConfiguration() và commitConfiguration()**.



* + Hàm **self.setupInputs():** được sử dụng để set up các **Input cho session**. Bây giờ biết thế thôi, tí đọc xuống dưới tự hiểu 😊))
  + **beginConfiguration():** Khi ta gọi hàm này, **session**  sẽ chuyển sang trạng thái **configuration mode**, trạng thái này sẽ cho phép ta thay đổi các cấu hình mà không ảnh hưởng đến trạng thái hiện tại. Điều đó có nghĩa là các **inputs và outputs** hiện tại vẫn sẽ được sử dụng trong khi ta đang cấu hình các thay đổi mới. Nếu ta ko sử dụng **beginConfiguratio(), thì mọi thay đổi sẽ được appy lập tức, điều này có thể dẫn đến các hành vi không mong muốn.**
  + **commitConfiguration():** Khi ta gọi hàm này, **session** sẽ updates các cấu hình sử dụng. Nếu ta ko sử dụng hàm này, **session** sẽ tiếp túc được vận hành với các cấu hình trước đó.
  + **startRunning():** Khi ta gọi phương thức này, **session** bắt đầu **capture data** từ input devices và xử lý dữ liệu đó. Một điều quan trọng là hàm này khi thực thi sẽ tốn rất nhiều thời gian nếu **session có nhiều inputs và outputs hoặc với các settings phức tạp** nên ta cần thực thi nó trong **background Thread.** Ta cũng nên gọi hàm **stopRunning()** để tránh tốn nhiều tài nguyên hệ thống và đỡ tiêu hao pin.
  + Ở đây, ta quan tâm về **presets(Các cài đặt trước),** cái này sẽ cấu hình cho chất lượng của **Output**. Mà **capture session** là người trung gian giữa inputs và outputs, nên nó cũng cung cấp các tính năng cho ta thực thi điều này. **AVCaptureSession.Presets** cung cấp cho ta các chức năng để tinh chỉnh chất lượng Outputs.



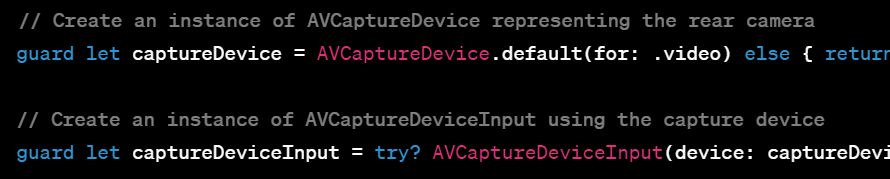
* + - Ta có đoạn code đầu sẽ kiếm tra xem **session có** thể support **.photo preset hay không**, nếu có, nó sẽ xét **.photo.** Điều này sẽ làm cho chất lượng hình ảnh, video từ camera sẽ trở lên tốt hơn, chất lượng hơn.
    - **automaticallyConfiguresCaptureDeviceForWideColor:** Ta sẽ cấu hình **session** tự điều chỉnh **color space và color range** cho inputs và outputs dữa trên khả năng của thiết bị.
    - Ta cũng có thể cấu hình thêm các thông tin như: **aspect retio, Depth Data**, **Zoom, Flash, White Balance, IOS, Color spaces, Tone Mapping,…**

### 2.2.2 Setting Up Inputs

**AVCaptureDevice:** là 1 class trong AVFoundation, nó đại diện cho các thiết bị vật lý được sử dụng để **capture** như camera hoặc microphone.

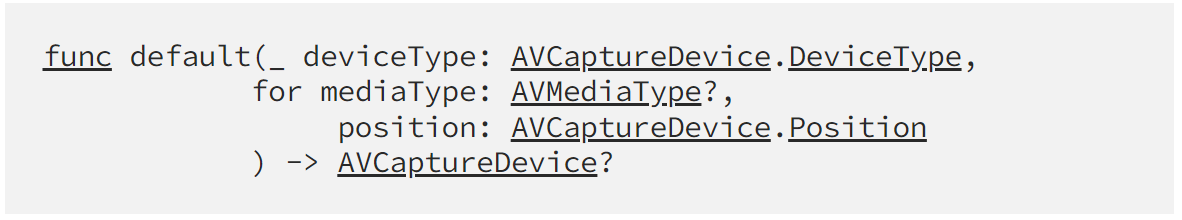
**AVCaptureDeviceInput**: đại diện cho một **input device** như là camera hoặc microphone mà có thể được thêm bởi session**.** Khi ta tạo 1 input(**AVCaptureDeviceInput**), ta phải chỉ định **nó phải là kiểu AVCaptureDevice nào.** Hiểu đơn giản là **AVCaptureDevice là** một thiết bị vật lý, nó thể là camera trước, camera sau, hoặc là microphone. **AVCaptureDeviceInput** là **input device**, nó là thằng nhận dữ liệu từ 1 thằng Instance **AVCaptureDevice mà ta gán cho nó**.

Hiểu tổng quát là ta tạo 1 Instance Input thuộc kiểu **AVCaptureDeviceInput,** thằng Input này sẽ nhận dữ liệu từ thằng thiết bị cụ thể ta gán cho nó, và thằng thiết bị cụ thể này thuộc kiểu **AVCaptureDevice.** Dữ liệu từ **device** tới **Input**, sau đó **Input** sẽ truyền dữ liệu vào cho **session để được output mong muốn.**



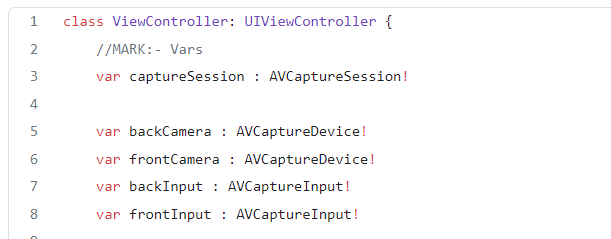
Ta có **2 options** để khai báo 1 Instance của AVCaptureDevice:

* Kiểu 1: **AVCaptureDevice.default(for: AVMediaType).** Trong đó AVMediaType có rất nhiều kiểu như là
* audio
* video
* text
* metadata
* Subtitle
* Text
* Ta có thể thấy struct này không chỉ dành cho **AVCaptureDevice,** nó bao gồm cả **subtile và text**. Tuy nhiên ta đang muốn làm việc với video, các thông tin như camera trước hoặc sau, nhưng ở đây chỉ cung cấp mỗi video, nên ta sẽ không sử dụng option này. Và chú ý ở đây ko có **photo.** Bởi vì camera không được bật khi ta cố chụp ảnh, mà nó đã được bật sẵn trước đó rồi. Chụp ảnh thực chất chỉ là lấy 1 frame ảnh từ **video mà thôi.**
* Kiểu 2:



* Kiểu 2 cung cấp thêm 2 parameter mới, đó là **position(front, back or unspecified) và a device type.**
  + **Device Type** có thể có các option là
    - builtInDualCamera
    - builtInDualWideCamera
    - buildInTripleCamera
    - **builtInWideAngleCamera**
    - builtInUltraWideCamera
    - builtInTelephotoCamera
    - builtInTrueDepthCamera
    - builtInMicrophone
    - externalUnkown
    - Ta thấy rằng ở đây có hết những kiểu thiết bị như **microphone, camera.** Ta biết rằng mọi thiết bị đều có **builtInWideAngleCamera, kiểu này hỗ trợ cả Camera trước và sau.**

Ví dụ Setup:



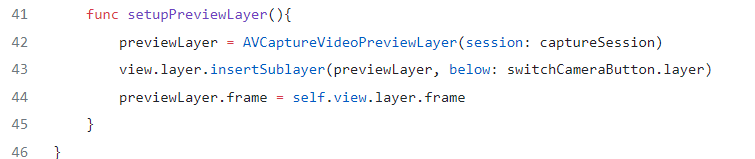


Ở ví dụ này, trong hàm **setupInputs(),** ta tạo ra các **Inputs** được gắn với các **camera sau**, sau đó add vào **session.** Lưu ý add cả trước và sau camera vào thì lỗi lòi mồm, vì tại 1 thời điểm session chỉ xử lý camera trước or camera sau.

### 2.2.3 Displaying The camera Feed

Ơ phần trên, ta đã có inputs, điều đó có nghĩa rằng hiện nay **capture session** đang nhận đượcvideo streams từ camera. AVFoundation cung cấp cho ta 1 cách đơn giản để có thể hiển thị video đó là sử dụng **AVCaptureVideoPreviewLayer.**

**The preview Layer** chỉ đơn giản là 1 **CALayer** mà ta có thể tạo ra từ **1 capture session,** và thêm nó như là 1 **sublayer vào view** của chúng ta.



### 2.2.4 Setting Up The Output và Taking a Picture

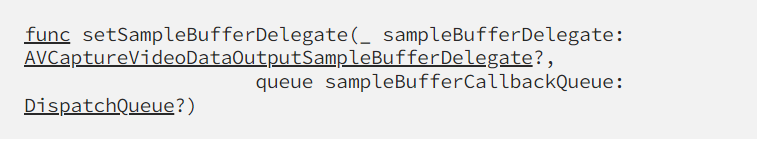
Trong phần trước, chúng ta đã khám phá ra **preview layer**. Đó cũng là 1 output.

Ta có 2 options, và cả 2 đều là object của **AVCaptureOutputs.** Nó sẽ cung cấp cho ta dữ liệu mà **capture session** nắm vai trò trung gian từ input thiết bị**.**

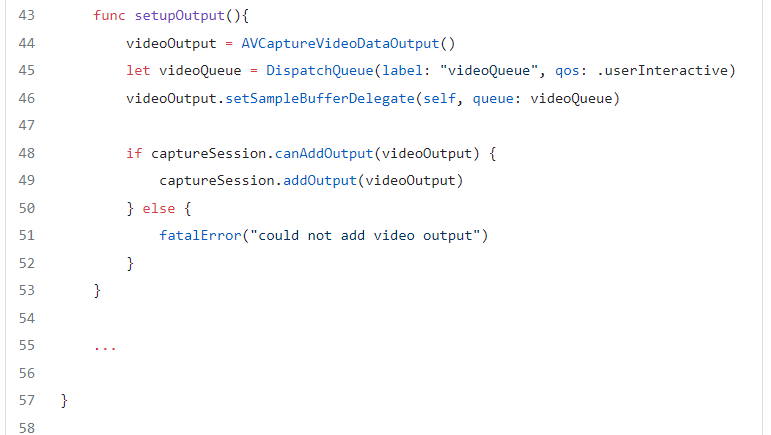
**Option 1**: Đây là 1 super simple option, được gọi là **AVCapturePhotoOutput**. Tất cả những gì ta cần làm là tạo object, đính nó với **session.** Sau đó khi người dùng nhất nút **capture button**, thì ta sẽ gọi **capturePhoto(with: delegate:),** từ đây ta đã nhận được 1 **photo object, từ đó ta có thể lưu ảnh.**

***Option 2***: Đây là 1 option mà sẽ trả lại **raw video frames.** Option này cho phép ta dễ dàng điều chỉnh hướng chụp ảnh. **Vậy ta sẽ sử dụng option này.**

**AVCaptureVideoDataOutput:** Là 1 capture output mà sẽ ghi lại video và cung cấp quyền truy cập vào các khung hình(frame) video để xử lý. Có 1 hàm quan trọng sau để setting output:



* **AVCaptureVideoDataOutputSampleBufferDelegate**: Đây là đối tượng sẽ nhận **captured sample buffers.** “**Captured sample buffers**” là các **raw data** được sinh ra bởi **capture device** khi quay video. Nó đại diện cho 1 sequence frames hoặc images mà được captured từ camera.
* **sampleBufferCallbackQueue:** đây là queue mà **callback** sẽ được thực thi trên đó.

**** ****

Ta thấy rằng việc setting output rất đơn giản 😊)) Chỉ định Queue để thực thi **callback,** và đối tượng nhận frame là **self 😊))**

Khi có sample buffer được capture, **AVCaptureVideoDataOutput** sẽ gọi phương thức **captureOutput(\_ output: AVCaptureOutput, didOutput sampleBuffer: CMSampleBuffer, from connection: AVCaptureConnection).** Từ phương thức này, ta có thể truy cập các **raw sample buffer** thông qua parameter thứ 2 là **CMSampleBuffer.**

**CMSampleBuffer** là 1 **Core Media** mà đại diện cho từng frame của audio hoặc video data, cùng với các thông tin metadata. **CMSampleBuffer** sẽ chứa các thông tin của audio và video như **timestamp, duration, … CMSampleBuffer** được sử dụng để di chuyển các **media sample data** thông qua **media pipeline.** Về cơ bản 1 **CMSampleBuffer** chứa rất nhiều các mảng dữ liệu khác nhau, những gì ta quan tâm bây giờ là 1 **image buffer.** Core Media cung cấp cho ta phương thức **CMSampleBufferGetImageBuffer**(). Hàm này sẽ return cho ta **CVImageBuffer.** Từ **image buffer** này, ta có thể lấy được **CIImage,** và từ đó ta có thể tạo được các **UIImage**.

* **CIImage:** là 1 **Core Image Data Type**, được apple cung cấp để lập trình viên dễ dàng làm việc với các images hơn với cách flexible và powerful way. Nó như là 1 container, có thế giữ bất kì kiểu image nào, nó rất hữu dụng khi mong muốn sử dụng các tính năng như điều chỉnh mờ hoặc màu cho hình ảnh. **CIImage** không bị rang buộc với bất kì kiểu hình ảnh nào hoặc độ phân giải pixel cụ thể nào, điều đó có nghĩa là ta có thể làm việc với bất kì loại dữ liệu hình ảnh nào, cho dù đó là ảnh, ảnh chụp màn hình hay bất kì raw pixel nào. Có 1 điều quan trọng **CIImage** là **immutable**. Vậy để có thể thay đổi là sử dụng **filter lên CGImage** để tạo mới 1 image.



* Ở đây ta có 1 button, khi nhấn button thì biến **takePicture** sẽ chuyển sang true
* Khi ta quay video, nó sẽ liên tục vào hàm **captureOutput**, vậy để khi nào nhấn mới chụp thì ta cần thêm biến **takePicture.**
* Thông qua **CMSampleBufferGetImageBuffer()** mà ta lấy được frame chụp được, và trích xuất để có ảnh.

### 2.2.5 Setting orientation

Ở phần trên, ta đã lấy được ảnh rồi, nhưng ta vẫn chưa lấy được đúng hướng ảnh. Trong khi **video preview layer** luôn hiển thị đúng hướng, nhưng data từ **AVCapture VideoDataOutput** thì không đúng. Chúng ta có thể fix điều này ở 2 nơi, đầu tiên ở phần kết nối giữa **output object và session** hoặc khi tạo UIImage từ CIImage.

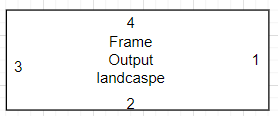
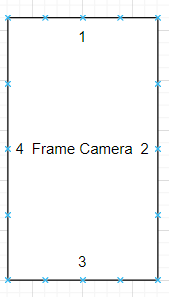
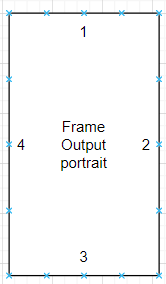
**AVCaptureConnection:** là kết nối giữa **capture input và capture output object ở capture session.** Connection cũng chỉ là các Object. Ta ngầm định tạo ra nó thông qua việc **addInput() và addOuput()** của **capture session.** Một đối tượng **connection object** có thể được truy cập từ mọi nơi thông qua **data pineline(**inputs, capture session và outputs**).** Từ document, chúng ta có thể cấu hình connection thông qua cấu hình thuộc tính **videoOrientation** của **output connections**

****

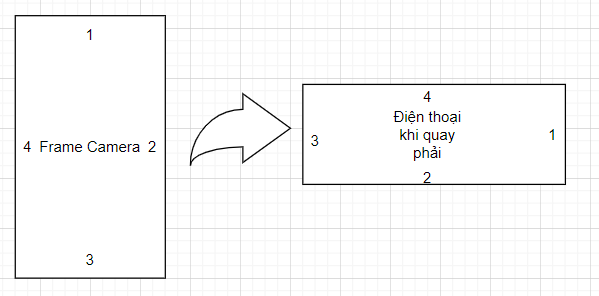
**.videoOrientation**: Là một biến thuộc kiểu enum **AVCaptureVideoOrientation. Enum đó gồm các case sau:**

* **portrait(Chân dung):** Ý là khi ta chụp chân dung, thì ta thường để dọc điện thoại, thì cái này cho phép ta thu được ảnh theo kiểu chiều dọc
* **portraitUpsideDown:**

Khi chụp ảnh, ta có 2 frame, frame camera và frame output mong muốn:

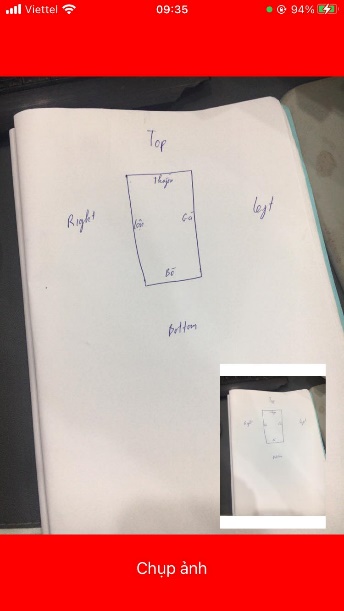
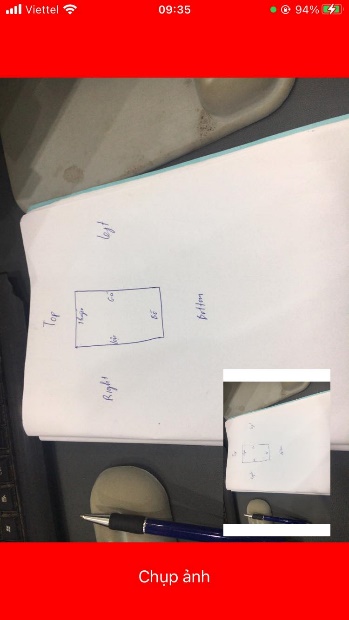


Đầu tiên, đối với các frame Output mong muốn, **vị trí của các điểm 1, 2, 3, 4 luôn không đổi**. Còn đối với khi xoay điện thoại**. frame camera cũng xoay luôn, dẫn đến vị trí điểm 1 2 3 4 của ảnh thu được cũng xoay như hình bên dưới.**



Từ **các điểm 1 2 3 4** cố định của **frame output cố định,** ta sẽ **phải xoay ảnh nhận được khi chụp theo đúng các điểm frame output cố định đó.**

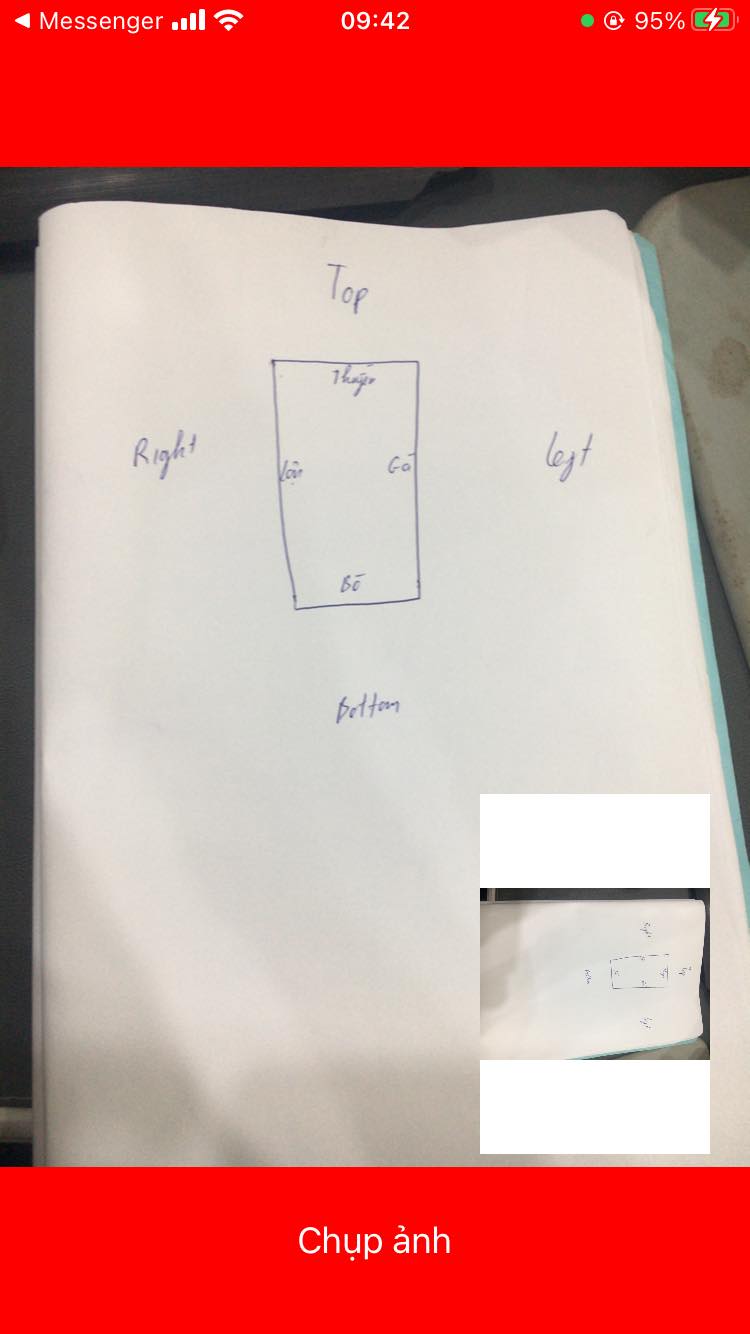
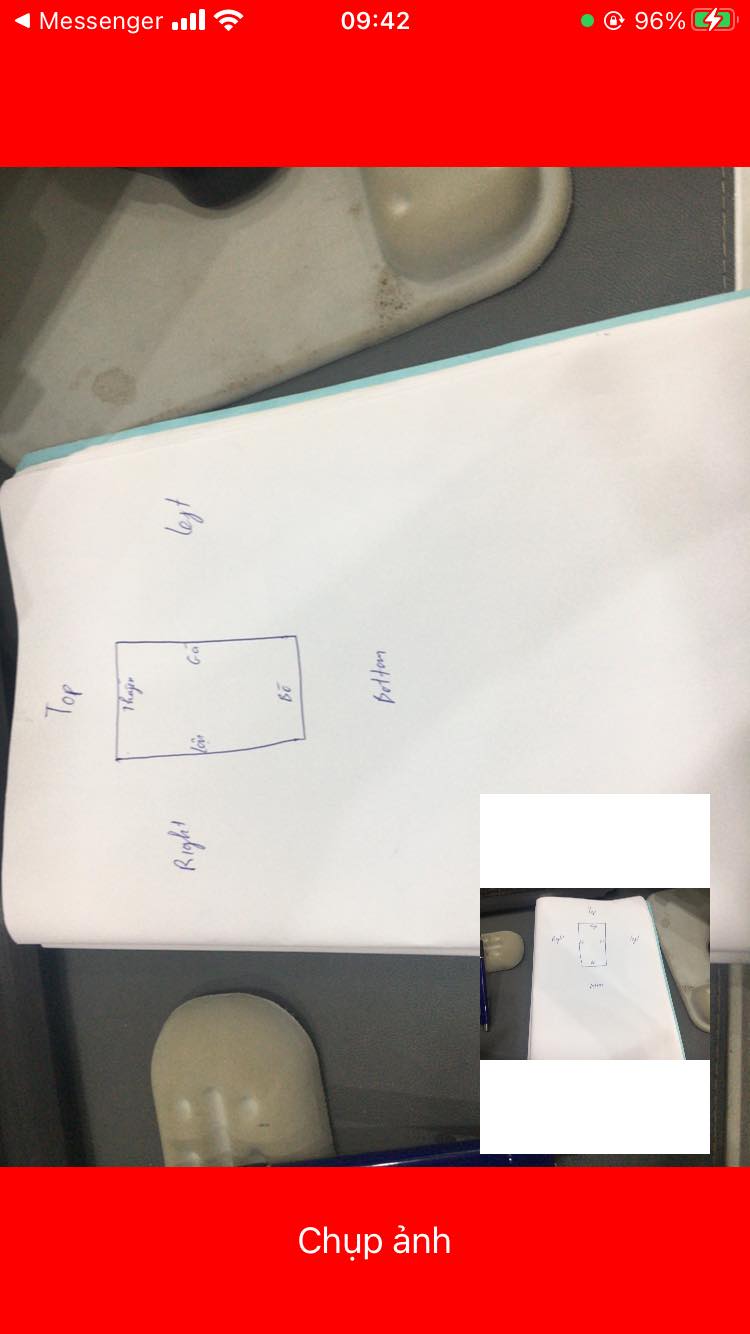
* **Với .portrait:** Điểm 1 luôn hướng ở **top**
* **Với .landscapeLeft:** Điểm 2 luôn ở **bottom**
* **Với .landscapeRight:** Điểm 4 luôn ở **bottom**
* **Với .portraitUpsideDown:** Điểm 4 luôn ở **top**

**VD: Với Portrait:**

Chụp thẳng: Quay phải điện thoại

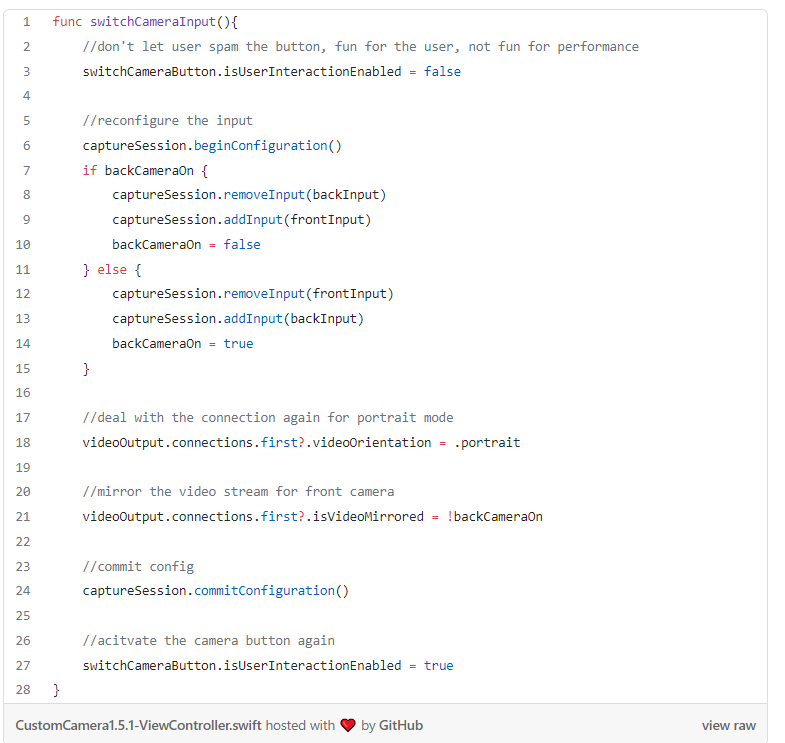
**Với .landscapeLeft**

Chụp thẳng: Quay phải điện thoại



### 2.2.6 Switching Cameras

Như ta đã biết, 1 output hiện tại chỉ có thể được kết nối với camera trước hoặc camera, vậy để đổi camera trước và sau cho nhau, ta phải **.remove() deviceInput trước đó, và thêm deviceInput mới vào.**



* Khi dùng camera trước, nó còn vấn đề mặt bị ngược hoặc giống với gương, để điểu chỉnh ta sử dụng **.isVideoOrientation .** Kiểu ta nhìn vào preview kiểu gì thì output như thế thì xét **.isVideoOrientation = true(Đối với camera sau).** Còn ta nhìn vào mà muốn ngược khi thì xét **.isVideoOrientation = false, lúc đó mặt ta bị lộn ngược lại mới đung**
* **Chú ý** ta cần xét lại**:**

