# Swift Study 10

2017. 02.04

# Swift 문법 및 ios

- dictionary / enum
- error handle (Error, try/catch, throws)
- AVAudioPlayer / AVAudioRecord / AVAudioSession
- UIImagePickerController

# Dictionary

- swift에 제공하는 mutable한 collection type 종류
- key:value라는 쌍데이터를 이루어져있으며, 적용할 타입제한이 없다.(단, 일치된 타입 사용)
- key의 타입 경우 해시연산 가능한 타입이어야 함 (식별가능한 키)

<형태>
[ key : value, key : value, …]

ex)
[ 1 : “aaa”, 2 : “bbb”, 3 : “ccc”]

<타입 선언>
Dictionary<key type : value type>

ex)
Dictionary<Int, String>

# Dictionary

<선언>
Dictionary<Int, String>()
[Int, String]()

<초기화>
let dic:Dictionary<Int, String> = [1 : “aaa”, 2 : “bbb”, 3 : “ccc”]
let dic = [ 1 : “aaa”, 2 : “bbb”, 3 : “ccc”]

<추가/수정/삭제>
var dic = [String:String]()

dic[“test1”] = “abc”  //test1라는 키이름으로 abc값 추가.
dic.updateValue(“test2”,forKey:”bbb”) //test2키의 값을 bbb로 수정(키값 존재하지 않을시 추가)
dic.updateValue(“test1”,forKey:”bbb”) //test1키의 값을 bbb로 수정
dic.removeValue(forKey: "test1")  //test1키의 키와 값을 삭제

# Dictionary

- array와 달리 순차적인 저장을 하지 않음 (key값을 기준으로 정렬, 순차적 저장순서X)

<순회탐색>

```
let dictionary = [“test2”:”bbb”, ”test1”:”aaa”]            // [String:String]()

for (key,value) in dictionary {
    print(key, value)
}
```

⇒ 출력

```
test1 bbb
test2 bbb
```

# Enum

- 열거형, 특정 주제에 관련 데이터를 멤버로 구성하기 위한 자료형 객체
- 남/여, 국가, 지역 등 구분 지어지는 데이터를 분류할 용도
- enum이라는 키워드로 선언하여 case별 값 카테고리 구분

<형태>

```
enum 열거형 이름 {
   case 멤버값
   case 멤버값
   case 멤버값
   case 멤버값
}
```

ex)

```
enum NATION {
   case korea
   case america
   case japan
   case china
}
```

# Enum

<특정값 적용>

```
enum NATION : String {
   case korea = “KR”
   case america = “EN”
   case japan = “JP”
   case china = “CN”
}
```

<사용>

```
NATION.korea
let nation:NATION = NATION.korea
let nation:NATION = .korea
```

<enum 특정값 사용>

```
NATION.korea.rawValue()   // => return “KR”
```

<활용>

```
switch nation{
  case .korea:  print(nation.rawValue())
  case .america: print(nation.rawValue())
  case .japan: print(nation.rawValue())
  case .china: print(nation.rawValue())
}
```

# Error handle - Error protocol

- 특정 조건에 대한 에러 제어 흐름을 제어하기 위한 문법적 장치
- 언어에서는 보통 exception (예외처리) 용어로 많이 사용
- Error라는 protocol타입을 구현한 enum타입에 에러를 정의
- Error protocol은 의미없는 빈 프로토콜로 표시의 의미가 강함.

```
public protocol Error {
}
```

<형태>

```
enum 에러명 : Error {
   case 에러함수명
   case 에러함수명(매개변수)
}
```

<형태>

```
enum IntegerParseError : Error{
   case nilNotParsing
   case characterNotParsing(char:Character)
}
```

# Error handle - throws / throw

- throws 키워드를 통해 에러 예외처리 호출정의
- throw로 작성한 에러를 호출 (에러를 던지다는 표현함)

```
<형태>
func 함수명(매개변수) throws -> 리턴형 {

    if 조건문 {
        //조건 실행
    } else {
        throw 에러명.에러함수
    }
    ...
}
```

```
ex)
func numCheck(value:Any?) throws ->
Int {
    if let num = value as? Int {
        return num
    } else {
        throw NumCheckError.notNum
    }
  }
```

# Error handle - do / try ~ catch

- throw로 던져지는 error를 호출받아 분기시키는 구문
- try로 throws에 대한 결과를 처리하여 실패시 catch로 에러처리

```
do {
    try expression
    statements
} catch pattern 1 {
    statements
} catch pattern 2 where condition {
    statements
}
```

ex)

```
func testNum(_ value:Any){
        do {
            let num = try numCheck(value)
            print(num)
         } catch {
            print((error as! NumCheckError).description)
         }
    }
```

# AVFoundation

- 영상 및 미디어에 대한 기능을 제공하는 라이브러리

`AVAudioPlayer`

의 인스턴스 `AVAudioPlayer`클래스는, 오디오 플레이어라는 파일 또는 메모리의 오디오 데이터의 재생을 제공합니다.

`AVAudioPlayerNode`

이 `AVAudioPlayerNode`클래스는 오디오 파일의 버퍼 또는 세그먼트를 재생합니다.

`AVAudioRecorder`

의 인스턴스 `AVAudioRecorder`오디오 레코더라는 클래스는 응용 프로그램에서 오디오 녹음 기능을 제공합니다. 오디오 레코더를 사용하면 다음을 수행할 수 있습니다.

`AVAudioSequencer`

`AVAudioSession`

오디오 세션은입니다 싱글 앱의 오디오 행동에 대한 당신의 의도를 앱 오디오 컨텍스트를 설정하는 데 사용하고 시스템에 표현하는 객체입니다.

# AVAudioPlayer

- AVFoundation 서브클래스로 음성재생에 대한 기능을 제공

AVAudioPlayer 객체 초기화

`init(contentsOf: URL)`

지정된 사운드 파일을 재생하기위한 오디오 플레이어를 초기화하고 반환합니다.

`init(data: Data)`

지정된 메모리 버퍼를 재생하기위한 오디오 플레이어를 초기화하고 반환합니다.

`init(contentsOf: URL, fileTypeHint: String?)`

지정된 URL 및 파일 유형 힌트를 사용하여 오디오 플레이어를 초기화하고 반환합니다.

`init(data: Data, fileTypeHint: String?)`

지정된 데이터 및 파일 형식 힌트를 사용하여 오디오 플레이어를 초기화하고 반환합니다.

# AVAudioPlayer

## 재생 구성 및 제어

`func play()`
비동기 적으로 사운드를 재생합니다.

`func play(atTime: TimeInterval)`
오디오 출력 장치의 타임 라인에서 지정된 지점에서 시작하여 비동기 적으로 사운드를 재생합니다.

`func pause()`
재생을 일시 중지합니다. 사운드는 중단 된 부분부터 다시 재생할 준비가됩니다.

`func stop()`
재생을 중지하고 재생에 필요한 설정을 취소합니다.

`func prepareToPlay()`
버퍼를 미리로드하여 재생할 오디오 플레이어를 준비합니다.

`var isPlaying: Bool`
오디오 플레이어가 재생 (여부를 나타내는 부울 값 `true`) 또는하지 ( `false`).

`var volume: Float`
에 이르기까지 오디오 플레이어의 재생 볼륨 `0.0`을 통해 `1.0`선형 규모.

# AVAudioPlayer

```
var rate: Float
```
오디오 플레이어의 재생 속도.

```
var enableRate: Bool
```
오디오 플레이어에서 재생 속도 조정을 사용할지 여부를 지정하는 부울 값

```
var numberOfLoops: Int
```
재생이 끝날 때까지 사운드가 처음으로 되돌아 오는 횟수입니다.

```
var delegate: AVAudioPlayerDelegate?
```
오디오 플레이어의 델리게이트 객체입니다.

```
var settings: [String : Any]
```
오디오 플레이어의 설정 사전. 플레이어와 관련된 사운드에 대한 정보가 들어 있습니다.

# AVAudioPlayer

## 사운드 정보 관리

`var numberOfChannels: Int`
오디오 플레이어와 관련된 사운드의 오디오 채널 수입니다.

`var channelAssignments: [AVAudioSessionChannelDescription]?`
의 배열 `AVAudioSessionChannelDescription`오디오 플레이어와 관련된 객체

`var duration: TimeInterval`
오디오 플레이어와 관련된 사운드의 총 지속 시간 (초)을 반환합니다.

`var currentTime: TimeInterval`
오디오 플레이어와 관련된 사운드 타임 라인 내의 재생 지점 (초)입니다.

`var deviceCurrentTime: TimeInterval`
오디오 출력 장치의 시간 값 (초)입니다.

`var url: URL?`
오디오 플레이어와 연결된 사운드의 URL입니다.

`var data: Data?`
오디오 플레이어와 연결된 사운드가 포함 된 데이터 객체입니다.

# AVAudioPlayer

## 오디오 레벨 미터링 사용하기

```
var isMeteringEnabled: Bool
```

오디오 플레이어의 오디오 수준 미터링 켜기 / 끄기 상태를 지정하는 부울 값입니다.

```
func averagePower(forChannel: Int)
```

연주되고있는 사운드의, 지정된 채널의 평균 전력 (데시벨)을 리턴합니다.

```
func peakPower(forChannel: Int)
```

지정된 채널의 최대 전력 (데시벨)을 재생중인 사운드에 대해 반환합니다.

```
func updateMeters()
```

오디오 플레이어의 모든 채널에 대한 평균 및 최대 전력 값을 새로 고칩니다.

# AVAudioRecord

| | |
|---|---|
| AVAudioRecorder 객체 초기화하기 | `init(url: URL, settings: [String : Any])`<br>오디오 레코더를 초기화하고 반환합니다. |
| 녹음 구성 및 제어 | `func prepareToRecord()`<br>오디오 파일을 생성하고 녹음을 위해 시스템을 준비합니다.<br><br>`func record()`<br>녹음을 시작하거나 다시 시작합니다.<br><br>`func record(atTime: TimeInterval)`<br>특정 시간에 녹음을 시작합니다.<br><br>`func record(forDuration: TimeInterval)`<br>지정된 시간 동안 기록합니다.<br><br>`func record(atTime: TimeInterval, forDuration: TimeInterval)`<br>지정된 시간 동안 지정된 시간에 녹음을 시작합니다. |

# AVAudioRecord

```
func pause()
```
녹음을 일시 중지합니다.

```
func stop()
```
녹음을 중지하고 오디오 파일을 닫습니다.

```
var delegate: AVAudioRecorderDelegate?
```
오디오 레코더 용의 위양 객체입니다.

```
func deleteRecording()
```
녹음 된 오디오 파일을 삭제합니다.

# AVAudioRecord

## 녹음 정보 관리

```
var isRecording: Bool
```
오디오 레코더가 녹음 중인지 여부를 나타내는 부울 값입니다.

```
var url: URL
```
오디오 레코더와 연결된 오디오 파일의 URL입니다.

```
var channelAssignments: [AVAudioSessionChannelDescription]?
```
의 배열 `AVAudioSessionChannelDescription`레코더와 관련된 개체입니다.

```
var currentTime: TimeInterval
```
기록 시작 이후의 시간 (초)입니다.

```
var deviceCurrentTime: TimeInterval
```
오디오 레코더가있는 호스트 장치의 시간 (초).

```
var settings: [String : Any]
```
오디오 레코더의 오디오 설정입니다.

# AVAudioRecord

## 오디오 레벨 미터링 사용하기

```
var isMeteringEnabled: Bool
```

오디오 레벨 미터링 사용 여부를 나타내는 부울 값입니다.

```
func updateMeters()
```

오디오 레코더의 모든 채널에 대한 평균 및 최대 전력 값을 새로 고칩니다.

```
func peakPower(forChannel: Int)
```

지정된 채널의 최대 전력 (데시벨)을 녹음 사운드에 대해 반환합니다.

```
func averagePower(forChannel: Int)
```

지정된 채널의 데시벨 (dB) 단위의, 녹음 중의 사운드의 평균 파워를 리턴합니다.

---

## 초기화 장치

```
init(url: URL, format: AVAudioFormat)
```

# AVAudioSession

- 다른앱 사용중 음성에 대한 제어를 관리하는 세션 (싱글턴 객체)

| | |
|---|---|
| 공유 오디오 세션 가져 오기 | `class func sharedInstance()`<br>싱글 톤 오디오 세션을 리턴합니다. |
| 기록 허가 요청 | `func requestRecordPermission(: @escaping PermissionBlock)`<br>오디오 녹음에 대한 사용자의 권한을 요청합니다.<br><br>`func recordPermission()`<br>현재 녹음 허용 상태입니다. |

# AVAudioSession

오디오 세션 관리

```
var category: String
```
현재 오디오 세션 카테고리입니다.

```
var availableCategories: [String]
```
장치에서 사용할 수있는 오디오 세션 범주입니다.

```
var categoryOptions: AVAudioSessionCategoryOptions
```
현재 오디오 세션 카테고리와 관련된 옵션입니다.

```
func setCategory(String)
```
현재의 오디오 세션 카테고리를 설정합니다.

```
func setCategory(String, with: AVAudioSessionCategoryOptions = [])
```
지정된 옵션을 사용하여 오디오 세션 범주를 설정합니다.

```
var mode: String
```
현재 오디오 세션 모드입니다.

# AVAudioSession

```
var availableModes: [String]
```

장치에서 사용할 수있는 오디오 세션 모드.

```
func setMode(String)
```

오디오 세션 모드를 설정합니다.

```
func setActive(Bool)
```

앱의 오디오 세션을 활성화하거나 비활성화합니다.

```
func setActive(Bool, with: AVAudioSessionSetActi
veOptions = [])
```

지정된 옵션을 사용하여 앱의 오디오 세션을 활성화하거나 비활성화합니다.

# Audio Session Categories

```
let AVAudioSessionCategoryAmbient: String
```
사운드 재생이 기본이 아닌 앱의 카테고리입니다. 즉, 앱을 사용 중지 한 상태에서 사운드를 성공적으로 사용할 수 있습니다.

```
let AVAudioSessionCategorySoloAmbient: String
```
당신이 가진 카테고리 설정하지 않으면 기본 범주에 대한 범주는, 사용 `setCategory(_:)`또는 `setCategory(_:with:)`방법을.

```
let AVAudioSessionCategoryPlayback: String
```
앱을 성공적으로 사용하기 위해 녹음 된 음악이나 기타 사운드를 재생하는 카테고리입니다.

```
let AVAudioSessionCategoryRecord: String
```
오디오 녹음 카테고리. 이 범주는 재생 오디오를 음소거합니다.

# Audio Session Categories

`let` **AVAudioSessionCategoryPlayAndRecord**: `String`

VoIP (Voice over Internet Protocol) 응용 프로그램과 같이 오디오 녹음 (입력) 및 재생 (출력) 범주입니다.

~~`let` **AVAudioSessionCategoryAudioProcessing**: `String`~~

오디오를 재생하거나 녹음하지 않는 동안 오디오 하드웨어 코덱 또는 신호 프로세서를 사용하기위한 범주입니다. 예를 들어 오프라인 오디오 형식 변환을 수행 할 때이 범주를 사용하십시오.

더 이상 사용되지 않는

`let` **AVAudioSessionCategoryMultiRoute**: `String`

오디오 데이터의 다른 스트림을 다른 출력 장치에 동시에 라우팅하는 범주입니다. 예를 들어이 범주를 사용하여 오디오를 USB 장치와 헤드폰 세트로 라우팅합니다. 이 범주를 사용하려면 사용 가능한 오디오 경로의 기능에 대한 자세한 지식과 상호 작용이 필요합니다.

# UIImagePickerController

- 사진/동영상 기능을 제공하는 사용자 인터페이스

피커 소스 설정

```
class func availableMediaTypes(for: UIImagePickerControllerSourceType)
```

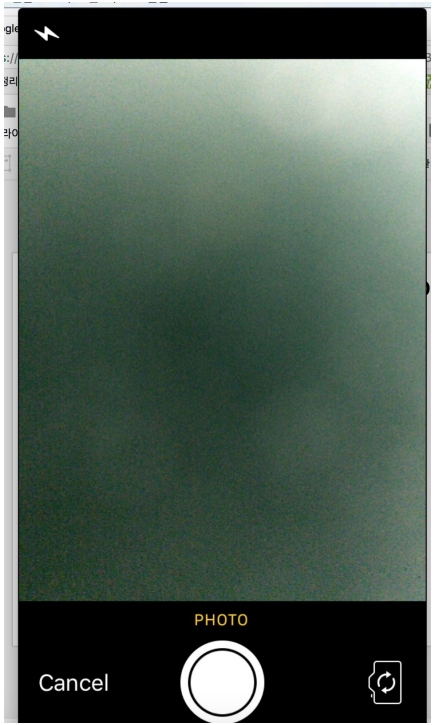
지정된 소스 유형에 사용할 수있는 미디어 유형의 배열을 반환합니다.

```
class func isSourceTypeAvailable(UIImagePickerControllerSourceType)
```

장치가 지정된 소스 유형을 사용하여 미디어를 선택할 수 있는지 여부를 나타내는 부울 값을 반환합니다.

```
var sourceType: UIImagePickerControllerSourceType
```

컨트롤러가 표시 할 피커 인터페이스 유형입니다.



# UIImagePickerControllerSourceType

`case photoLibrary`

장치의 사진 라이브러리를 이미지 선택 컨트롤러의 원본으로 지정합니다.

`case camera`

장치의 내장 카메라를 이미지 선택 컨트롤러의 소스로 지정합니다. 사용하여 (사용 가능한 등의 전면 또는 후면) 당신이 원하는 특정 카메라 나타내는 `cameraDevice`속성입니다.

`case savedPhotosAlbum`

장치의 카메라 롤 앨범을 이미지 선택 컨트롤러의 소스로 지정합니다. 장치에 카메라가 없으면 저장된 사진 앨범을 소스로 지정합니다.

# UIImagePickerController

## 선택기 구성

```
var allowsEditing: Bool
```
사용자가 선택한 스틸 이미지 또는 동영상을 편집 할 수 있는지 여부를 나타내는 부울 값입니다.

```
var delegate: (UIImagePickerControllerDelegate & UINavigationControllerDelegate)?
```
이미지 피커의 위임 객체입니다.

```
var mediaTypes: [String]
```
미디어 선택 컨트롤러가 액세스 할 미디어 유형을 나타내는 배열입니다.

---

## 비디오 캡처 옵션 구성하기

```
var videoQuality: UIImagePickerControllerQualityType
```
비디오 녹화 및 트랜스 코딩 품질.

```
var videoMaximumDuration: TimeInterval
```
비디오 녹화의 최대 지속 시간 (초).

# UIImagePickerController

## 카메라 컨트롤 사용자 정의하기

```
var showsCameraControls: Bool
```
이미지 선택기에 기본 카메라 컨트롤이 표시되는지 여부를 나타냅니다.

```
var cameraOverlayView: UIView?
```
기본 이미지 선택기 인터페이스 맨 위에 표시 할보기입니다.

```
var cameraViewTransform: CGAffineTransform
```
카메라의 미리보기 이미지에 적용 할 변형입니다.

## 스틸 이미지 또는 동영상 캡처

```
func takePicture()
```
카메라를 사용하여 정지 이미지를 캡처합니다.

```
func startVideoCapture()
```
에 의해 지정된 카메라를 사용하여 비디오 캡처 시작
`UIImagePickerControllerCameraDevice`속성을.

```
func stopVideoCapture()
```
비디오 캡처를 중단합니다.

# UIImagePickerController

## 카메라 구성

```
var cameraDevice: UIImagePickerControllerCameraDevice
```
이미지 선택 컨트롤러가 사용하는 카메라.

```
class func isCameraDeviceAvailable(UIImagePickerControllerCameraDevice)
```
지정된 카메라가 사용 가능한지 여부를 나타내는 부울 값을 반환합니다.

```
class func availableCaptureModes(for: UIImagePickerControllerCameraDevice)
```
배열 돌려 `NSNumber`소정의 카메라 장치에 의해 지원되는 촬영 모드를 나타내는 개체.

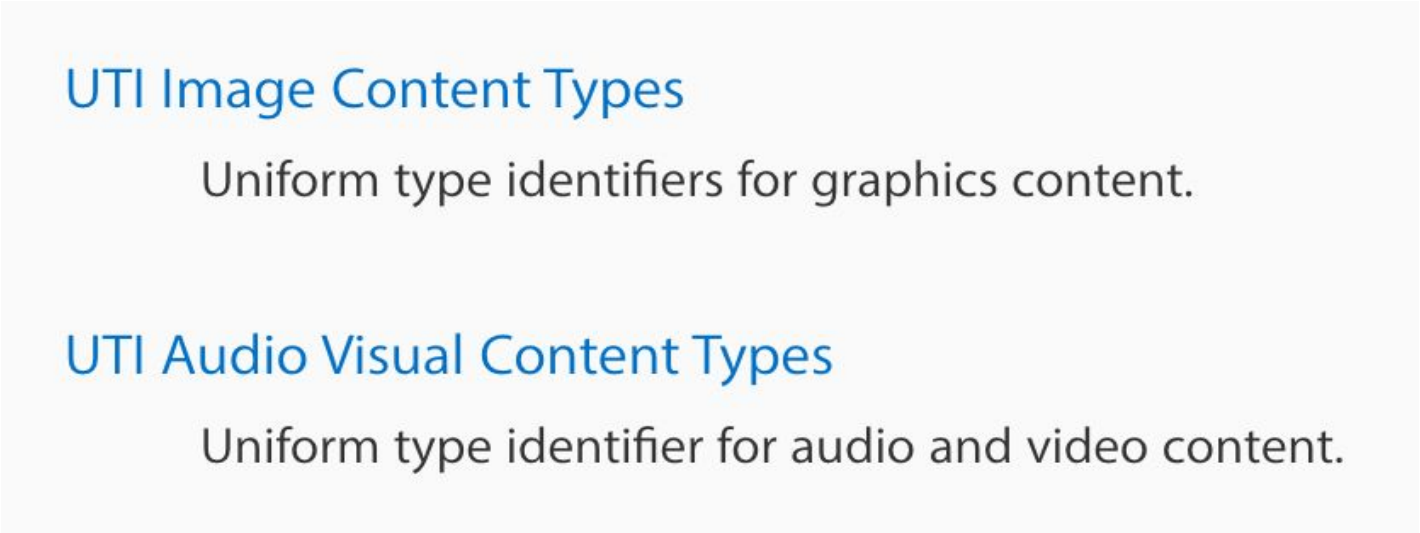
```
var cameraCaptureMode: UIImagePickerControllerCameraCaptureMode
```
카메라가 사용하는 캡처 모드.

```
var cameraFlashMode: UIImagePickerControllerCameraFlashMode
```
활성 카메라에서 사용하는 플래시 모드.

```
class func isFlashAvailable(for: UIImagePickerControllerCameraDevice)
```
지정된 카메라에 플래시 조명 기능이 있는지 여부를 나타냅니다.

# UTType

- Uniform Type Identifiers
- 파일 형식이나 메모리 내 데이터 형식, 디렉터리, 볼륨, 패키지와 같은 다른 종류의 엔터티 형식을 설명하는데도 사용
- MobileCoreServices 프레임워크 포함됨



# UTI Image Content Types

```
let kUTTypeImage: CFString
```
The abstract type identifier for image data.

```
let kUTTypeJPEG: CFString
```
The type identifier for a JPEG image.

```
let kUTTypeJPEG2000: CFString
```
The type identifier for a JPEG-2000 image.

```
let kUTTypeTIFF: CFString
```
The type identifier for a TIFF image.

```
let kUTTypePICT: CFString
```
The type identifier for a Quickdraw PICT.

```
let kUTTypeGIF: CFString
```

# UTI Audio Visual Content Types

```
let kUTTypeMovie: CFString
```
An abstract type identifier for a media format which m
what users would label a "movie"

```
let kUTTypeVideo: CFString
```
An abstract type identifier for pure video data(no audi

```
let kUTTypeAudio: CFString
```
An abstract type identifier for pure audio data (no vide

```
let kUTTypeQuickTimeMovie: CFString
```
The type identifier for a QuickTime movie.

```
let kUTTypeMPEG: CFString
```
The type identifier for a MPEG-1 or MPEG-2 movie.

```
let kUTTypeMPEG4: CFString
```