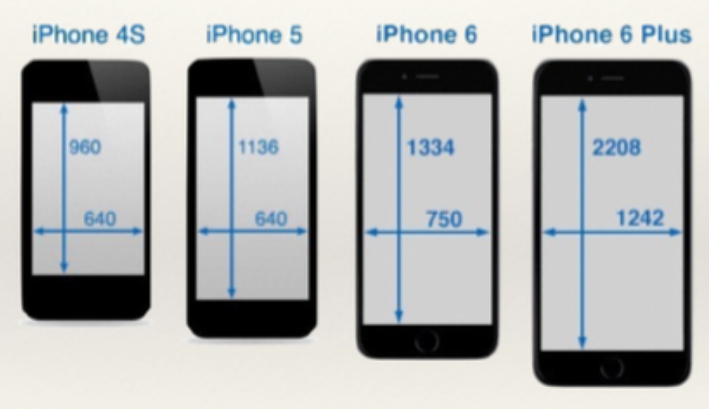
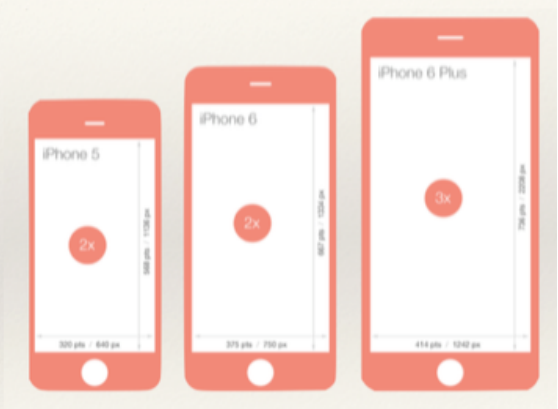
ios 입문 강의

8강 <UI 기본 컨포넌트>

1. UI Base Guide

(1) UI Base Guide





* UI에서는 포인트와 픽셀 (단위) 의 전환이 어떻게 되는지가 포인트

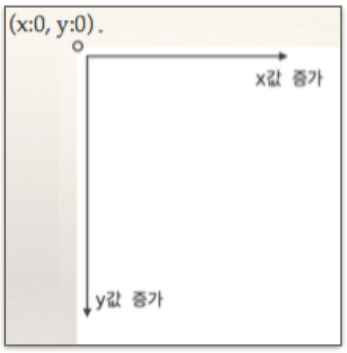
	Points	Pixels
Non Retina iPhone 1 point = 1 pixel	320x480	320x480
iPhone 4 (Retina Screen) 1 point = 4 pixels	320x480	640×960
iPhone 5 (Retina Screen) 1 point = 4 pixels	320x568	640×1136

  * 코딩할 때는 포인트 단위 / 디자인 할 때는 픽셀 단위
  * ⭐ 디자이너와의 협업 시, 디자이너는 픽셀로 주더라도 개발자는 무조건 포인트로 계산해야한다.
    * -> 디자인에서의 픽셀을 1/2로 줄여서 포인트로 바꾸는 것
* 6+에서는 1/3로 줄음

※ 참고

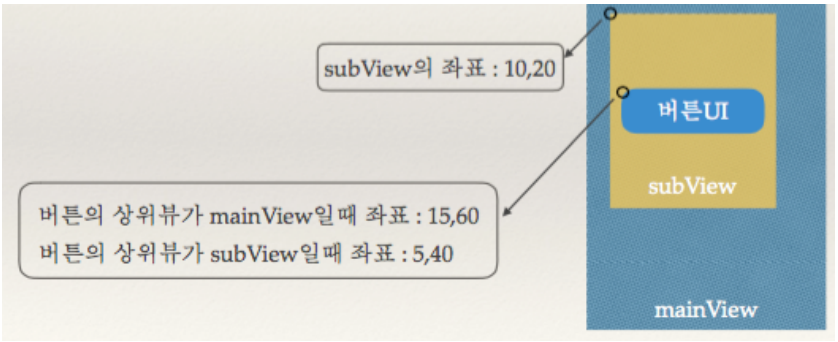
- UI뷰는 포토샵 처럼 생각하면 된다. 레이어를 하나씩 위로 쌓아 올라간다고 생각하면 된다.
- 참고 링크: http://i.stack.imgur.com/I9EXp.png

(2) 좌표계



* view 기준 좌측 상단이 (0,0)
  * -> 오른쪽으로 갈수록 x 값 증가 / 왼쪽 갈 수록 y값 증가

*** 뷰의 좌표는 상대 좌표이다**



* 뷰의 위치는 상대적으로 자신의 바로 위인 상위 뷰 기준으로 위치를 잡는다.
  * 버튼이 어떤 레이어 위에 있으면 그 레이어 좌측 상단을 0,0으로 잡고 좌표가 만들어진다.

(3) UIKit

* 코코아 터치 프레임워크에 추가된 UI관련 기능들의 모음
  * import UIKit -> UIKit라는 프레임워크를 우리가 가져다가 쓰겠다는 것이다.
  * 우리가 항상 썼던 UIViewController - 이것도 다 UIKit에 다 있는 것

* **프레임워크 / 라이브러리**
  * 프레임워크와 라이브러리 모두 클래스들의 모음
  * 유용한 내용의 누군가 만들어놓은 프로젝트를 배포한 것 -> 라이브러리
    * 대부분의 오픈소스는 라이브러리라고 할 수 있다.
  * 사용하기 쉬운 맥용 UI를 애플에서 제공하는 것이 UIKit이다.

* **라이브러리와 프레임워크의 차이**

특징	프레임워크	라이브러리
유저코드의 작성	프레임워크 클래스를 서브 클래싱 해서 작성	독립적으로 작성
호출흐름	프레임워크코드가 유저코드를 호출	유저코드가 라이브러리를 호출
실행흐름	프레임워크가 제어	유저코드가 제어
객체의 연동	구조프레임워크가 정의	독자적으로 정의

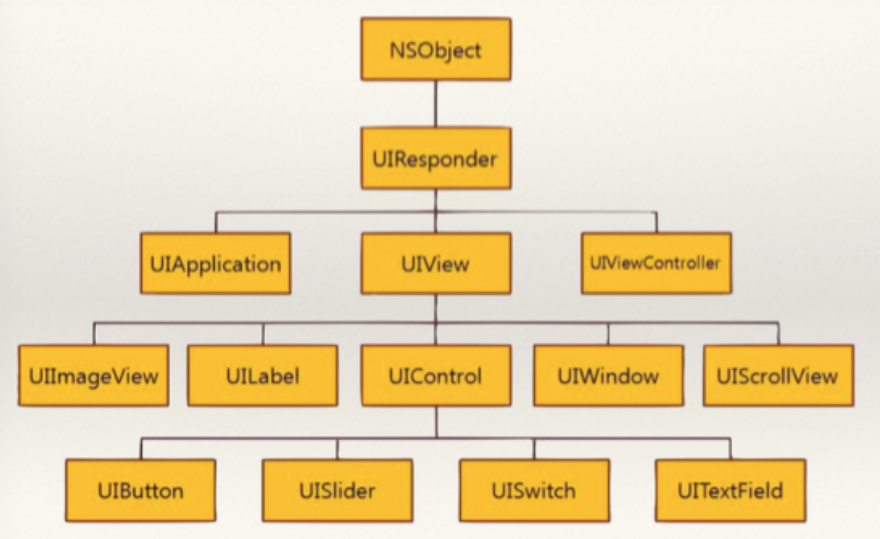
〈표 1〉 프레임워크와 라이브러리의 비교

출처 - 지디넷 코리아 [프레임워크 전략 ①] 프레임워크의 재발견

http://www.zdnet.co.kr/news/news_view.asp?artice_id=00000391160910&type=det&re=

  * 프레임워크
    * **일정한 룰을 통해** 뼈대를 상속받아서 추가적 개발
  * 라이브러리
    * **특화된 기능 단위로 쪼개져 가져다 쓰는 것**

(3) UIKit 계층도



* UIRespoder
  * 사용자의 반응을 관할

* UIView
  * 눈으로 보이는 뷰

* UIViewController
  * rootview를 꼭 하나씩 가지고 있다.
  * 그 rootview위에 올려질 View들을 관할

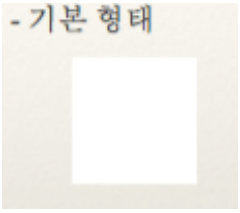
(3) UIResponder

* 응답에 관한 클래스 (포커싱)
  * 클래스는 속성(변수)과 메소드(행동)로 이루어져있다.
  * 우리가 써야할 용도에 맞춰 속성과 메소드를 가져다 쓰면 된다.

```
주요 요소

func becomeFirstResponder() -> Bool
func resignFirstResponder() -> Bool

func touchesBegan(touches: Set<NSObject>, withEvent event: UIEvent)
func touchesMoved(touches: Set<NSObject>, withEvent event: UIEvent)
func touchesEnded(touches: Set<NSObject>, withEvent event: UIEvent)
func touchesCancelled(touches: Set<NSObject>!, withEvent event: UIEvent!)
```

* becomFirstResopder
  * 마우스 커서 / 텍스트를 입력하는 UI
  * 키보드가 올라가면 값은 true가 된다.
* resignFirstResopder
  * 마우스 커서를 빼는 것

(4) UIView



* 기본형태 : 하얀 네모
* UIComponent들의 기본이 되는 View
* ios에서는 view들의 집합으로 화면이 구성된다

```
주요 요소

var frame: CGRect
var userInteractionEnabled: Bool
var tag: Int
var backgroundColor: UIColor?

init(frame: CGRect)
func addSubview(view: UIView)
func removeFromSuperview()
```

* 중요한 개념: frame
  * CGRect - size (가로/세로) / origin (x,y) 네가지 값으로 이루어진 데이터 타입
* userInteractionEnabled : Bool
  * 이게 true일때 UIRespoder에 있던 touchesBegan이 불림
  * (기본값) UIButton은 무조건 true <-> 이미지나 라벨은 false로 되어있음
    * UIImageview위에 button을 올리면-> 무조건 False가 됨

(기본값은 commend 누르고 해당 문서로 들어가면 확인 가능)

* tag
  * 여러 UI들을 구분하기 위해 달아주는 말 그대로 태그같은 것. Int 타입으로 숫자가 들어간다.

* init (frame: CGRect)
  * init할 때 frame을 넣었다.
  * 객체를 뷰로써 보이게끔 한 것 (뷰가 되려면 가로[x], 세로[y]가 있어야 한다. 이 때 설정해 줄 것.)

* addSubview
  * 내 뷰에 새로운 뷰를 올리면(addSubview하면) 원래 있던 내 뷰의 sub뷰가 만들어진다.
    * (바로 상위 것의 sub뷰)
  * addSubview는 부모가 자식을 소유하는 개념

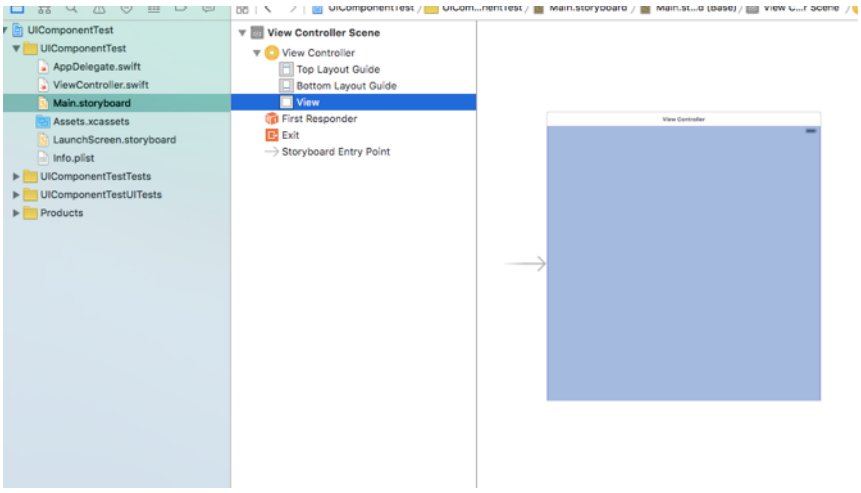
* removeFromSuperview
  * 자식이 부모를 떠나는 것

(예제 및 실습을 해보면 이해가 더 쉬워질 것입니다! 😊)

* 예제

```
//뷰 객체 생성 - 위치 설정
let secondView:UIView = UIView(frame: CGRect(x: 0, y: 0,
width: 100, height: 100))

//뷰 위에 서브뷰로 추가
self.view.addSubview(secondView)
//뷰의 배경색 설정
secondView.backgroundColor = UIColor.blueColor()
```

* **뷰 객체 생성 - 위치 생성**
  * frame의 초기화값을 넣음 (x, y - 위치 / width, height - 크기)
  * 기본 뷰는 UIViewcontroller - > 기본적으로 View가 하나씩 있다.(rootView)



IOS 입문 5기 화이팅!❤️

-> 이 rootView는 항상 전체 화면을 포함하고 있다.

* ※ 참고 : UI는 객체를 생성할 때 보통 var말고 let을 씀

* 뷰 위에 서브뷰로 추가
  * self.view.addSubview(secondView)
  * 상위뷰가 내 자식 뷰를 갖는 것
* secondview.backgroundColor
  * 배경 색상을 넣어줌

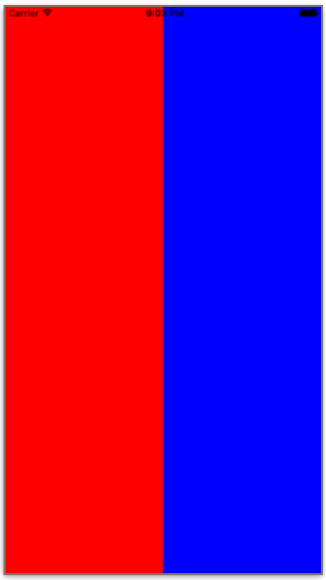
*** 순서**

1. 객체생성
2. 프레임 - 위치, 크기
3. addsubview로 추가 - 화면에 보이게끔 한다

* 각각의 것들이 어떤 속성을 가지는지
* 내가 쓸 것들이 무엇인지 찾아내는 것

실습 1

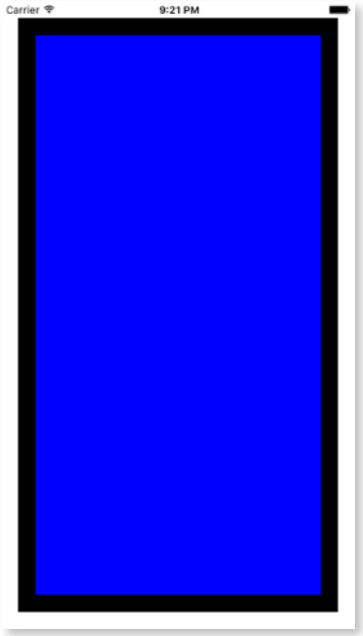
View 2개로 만들어보기!



```swift
/* 가로 길이 */
let myWidth:CGFloat = (self.view.frame.size.width / 2)

/* 세로 길이 */
let myHeight:CGFloat = (self.view.frame.size.height)

/* viewRect 설정 */
let redViewRect:CGRect = CGRect(x: 0, y: 0, width: myWidth, height: myHeight)

/* 왼쪽 View */
let redView:UIView = UIView(frame: redViewRect)
redView.backgroundColor = UIColor.redColor()
self.view.addSubview(redView)

/* 오른쪽 viewRect 설정 */
let blueViewRect:CGRect = CGRect(x: myWidth, y: 0, width: myWidth, height: myHeight)

/* 오른쪽 View */
let buleView:UIView = UIView(frame: blueViewRect)
buleView.backgroundColor = UIColor.blueColor()
self.view.addSubview(buleView)
```

실습 2

만들어보기!



자신의 부모 뷰에서 각각 20씩 떨어져있는 뷰 2개 입니다.

view들의 가로, 세로 길이는
자신 부모뷰와 40씩 차이가 날 것이다
(양 옆, 아래 위 magin이 20씩이니까)

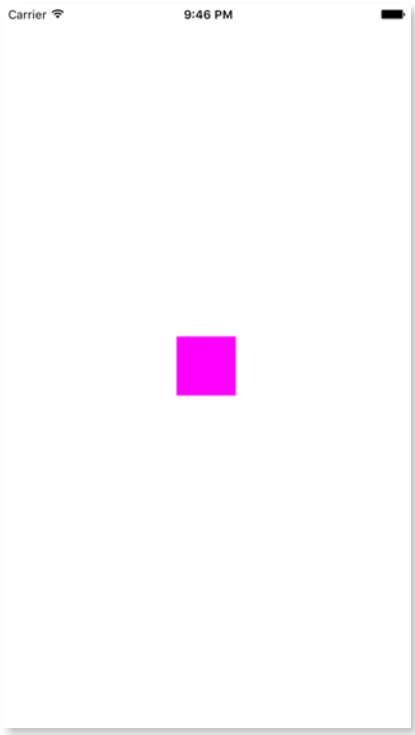
반복되는 마진 값을 변수로 두고
활용하면 추후 수정하기에 좋다.

```
/* magin */
let magin:CGFloat = 20

/* 가로 길이 */
let myWidth:CGFloat = (self.view.frame.size.width-magin*2)

/* 세로 길이 */
let myHeight:CGFloat = (self.view.frame.size.height-magin*2)

/* viewRect 설정 */
let blackViewRect:CGRect = CGRect(x:magin , y: magin, width: myWidth, height: myHeight)

/* 검은색 View */
let blackView:UIView = UIView(frame: blackViewRect)
blackView.backgroundColor = UIColor.blackColor()
self.view.addSubview(blackView)

/* 가로 길이 */
let myWidth2:CGFloat = (blackView.frame.size.width-magin*2)

/* 세로 길이 */
let myHeight2:CGFloat = (blackView.frame.size.height-magin*2)

/* viewRect 설정 */
let yellowViewRect:CGRect = CGRect(x:magin , y: magin, width: myWidth2, height: myHeight2)

/* 노란색 View */
let yellowView:UIView = UIView(frame: yellowViewRect)
yellowView.backgroundColor = UIColor.yellowColor()
blackView.addSubview(yellowView)
```

IOS 입문 5기 화이팅!❤️

실습 3

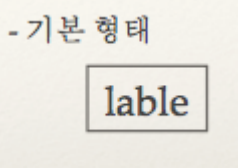
전체 뷰의 정 중앙에 위치한 뷰 만들어보기




```
// 사이즈 지정
let centerSize:CGSize = CGSize(width: 60, height: 60)

// x,y 좌표
let centerViewX = (self.view.frame.size.width/2 - centerSize.width/2)
let centerViewY = (self.view.frame.size.height/2 - centerSize.height/2)

// 좌표 및 사이즈 설정
let viewRect:CGRect = CGRect(x: centerViewX, y: centerViewY, width: centerSize.width,
                             height: centerSize.height)

/* 뷰 객체 만들고 위에 속성 및 컬러, subview 적용 */
let centerView:UIView = UIView(frame: viewRect)
centerView.backgroundColor = UIColor.magentaColor()
self.view.addSubview(centerView)
```

(5) UILable : UIView

* UIview를 상속받음
* 텍스트 정보를 표시해주는 UI



* 주요 요소
  * var text: String?
  * var font: UIFont!
  * var textColor: UIColor!
  * var numberOfLines: Int
  * var textAlignment: NSTextAlignment

* UILable - 예제

```
//레이블 객체 생성
let newLb:UILabel = UILabel(frame: CGRect(x: 0, y: 0, width:
100, height: 100))
//레이블 추가
self.view.addSubview(newLb)
//텍스트 입력
newLb.text = "텍스트 입력"
//텍스트 폰트
newLb.font = UIFont.systemFontOfSize(16)
//텍스트 컬러
newLb.textColor = UIColor.whiteColor()
//텍스트 정렬
newLb.textAlignment = NSTextAlignment.Left
```

실습 4

아래 화면을 만들어보기

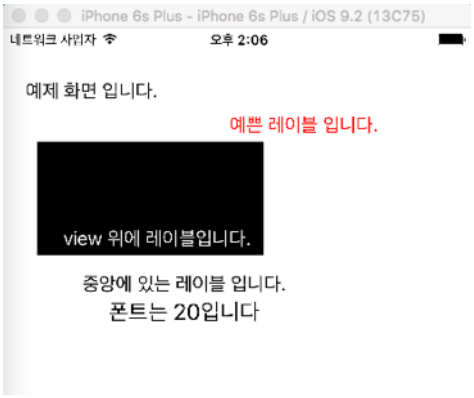

```
// 사이즈 지정
let centerSize:CGSize = CGSize(width: 150, height: 150)

// x,y 좌표
let centerViewX = (self.view.frame.size.width/2 - centerSize.width/2)
let centerViewY = (self.view.frame.size.height/2 - centerSize.height/2)

// 좌표 및 사이즈 설정
let viewRect:CGRect = CGRect(x: centerViewX, y: centerViewY, width: centerSize.width,
                             height: centerSize.height)

/* 뷰 객체 만들고 위에 속성 및 컬러, subview 적용 */
let centerView:UIView = UIView(frame: viewRect)
centerView.backgroundColor = UIColor.magentaColor()
self.view.addSubview(centerView)


// 좌표 및 사이즈 설정
let labelRect:CGRect = CGRect(x: 0, y: 0, width: centerSize.width,
                             height: centerSize.height)

/* 레이블 객체 만들고 위에 속성 및 컬러, subview 적용 */
let textLabel:UILabel = UILabel(frame: labelRect)
textLabel.text = "안녕하세요. \n swift 재미있죠? \n 하하하하"
textLabel.numberOfLines = 3
textLabel.textAlignment = NSTextAlignment.Center

centerView.addSubview(textLabel)
```

Label 코드 부분

IOS 입문 5기 화이팅!❤️

실습 5

아래 화면을 만들어보기



———————— 예제 답안 ————————

(레이블 위치 및 크기는 정해진 수치가 없으니 동일하지 않으셔도 됩니다.

단, 중요한 것은 객체를 만들고, addSubView, 텍스트, 텍스트 컬러, 중앙정렬을 적용시키는 것!)

```
// 까만 네모
let blackView:UIView = UIView(frame: CGRect(x: 30, y: 100, width: 200, height: 100)
self.view.addSubview(blackView) // 뷰에 올리기
blackView.backgroundColor = UIColor.blackColor()

// 예제화면입니다
let text1:UILabel = UILabel(frame: CGRect(x: 20, y: 50, width: 140, height: 10))
self.view.addSubview(text1)
text1.text = "예제 화면 입니다."
//text1.backgroundColor = UIColor.blueColor()

// 예쁜 레이블입니다
let text2:UILabel = UILabel(frame: CGRect(x: 200, y: 80, width: 140, height: 10))
self.view.addSubview(text2)
text2.text = "예쁜 레이블 입니다."
text2.textColor = UIColor.redColor()
//text2.backgroundColor = UIColor.blueColor()

//view 위에 레이블입니다
let text3:UILabel = UILabel(frame: CGRect(x: 53, y: 180, width: 180, height: 10))
self.view.addSubview(text3)
text3.text = "view 위에 레이블입니다."
text3.textColor = UIColor.whiteColor()
//text3.backgroundColor = UIColor.blueColor()

// 중앙에 있는 레이블 입니다
let text4:UILabel = UILabel(frame: CGRect(x: 60, y: 220, width: 200, height: 10))
self.view.addSubview(text4)
text4.text = "중앙에 있는 레이블 입니다."
//text4.backgroundColor = UIColor.blueColor()
text4.textAlignment = NSTextAlignment.Center

// 폰트는 20입니다.
let text5:UILabel = UILabel(frame: CGRect(x: 60, y: 240, width: 200, height: 20))
self.view.addSubview(text5)
text5.text = "폰트는 20입니다"
text5.font = UIFont.systemFontOfSize(20)
//text5.backgroundColor = UIColor.blueColor()
text5.textAlignment = NSTextAlignment.Center
```

투명한 배경의 라벨들은
처음에 Label 크기가 가늠이 되지 않으니,
임시로 배경 색깔을 넣어주었다가
이후 주석처리하면 됩니다.