CA ⇒ Core Animation

#CALayer

```
view.layer.backgroundColor = UIColor.blue.cgColor   // cgColor: core graph colors
view.backgroundColor = UIColor.blue                 // ↕ equivalent each other

// press Alt and hover over -> get help
view.layer.cornerRadius = CGFloat(10.0)
view.layer.borderWidth = CGFloat(5.0)
view.layer.borderColor = UIColor.orange.cgColor
view.layer.shadowOpacity = 0.7
view.layer.shadowRadius = CGFloat(15.0)
view.layer.shadowOffset = CGSize(width: CGFloat(-15), height: CGFloat(15))
view.layer.shadowOffset = CGSize.zero
view.layer.contents = UIImage(named: "cat")?.cgImage
view.layer.masksToBounds = true // will also clip the shadow

view.layer.contentsGravity = CALayerContentsGravity.center // center the image
```

// stand alone layer

```
lazy var newlayer: CALayer = {
        return CALayer()
}()

viewDidLoad
:

func setup

view.layer.addSublayer(standAloneLayer)

override func viewDidLayoutSubviews(){
  super.viewDidLayoutSubviews()
  newlayer.frame = view.bounds
}
```

Gradients with CAGradientLayer

```
import UIKit

class ViewController: UIViewController{

    lay var gradientLayer: CAGradientLayer ={
        return CAGradientLayer()
    }()

    override func viewDidLoad(){
        super.viewDidLoad()
        setGradientlayer()
    }

    override func viewDidLayoutSubviews(){
        super.viewDidLayoutSubviews()
        gradientLayer.frame = view.bounds
    }

    func setGradientLayer(){
        gradientLayer.colors = [UIColor.red.cgColor, UIColor.green.cgColor]
        gradientLayer.startPoint = CGPoint(x: 0.0, y: 0.0)
        gradientLayer.endPoint = CGPoint(x: 1.0, y: 1.0)
        view.layer.addSublayer(gradientLayer)
    }
```

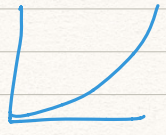
→ 110. not forget . cgColor ↖

Animation

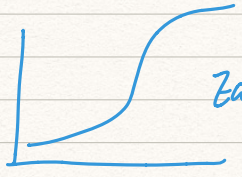
time + pacing + Spring

Timing functions ⇒ CAMediaTiming function

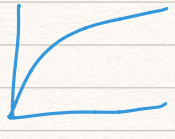
Linear



ease in



Ease In Out



ease out

UIView Animation

```
import UIKit

class ViewController: UIViewController{
    @IBOutlet weak var block: UIView!

    override func viewDidLoad(){
        super.viewDidLoad()

        let tap = UITapGestureRecognizer(target: self, action: #selector(blockTapped))
        block.addGestureRecognizer(tap)
        block.isUserInteractionEnabled = true
    }

    @objc func blockTapped(){
        UIView.animate(withDuration: 1.2){
            let sizeMultiplier: CGFloat = 1.5
            //let currentBlockFrame = self.block.frame
            //self.block.frame.size = CGSize(width: currentBlockFrame.width * sizeMultiplier,
                                             height: currentBlockFrame.height * sizeMultiplier)

            let newWidth = self.block.frame.width * sizeMultiplier
            let newHeight = self.block.frame.height * sizeMultiplier

            let newX = self.block.frame.origin.x - (newWidth - self.block.frame.size.width)/2
            let newY = self.block.frame.origin.y - (newHeight - self.block.frame.size.height)/2

            self.block.frame = CGRect(x: NewX, y: newY, width: newWidth, height: newHeight)

        }
    }
```

How to chain animations

```
import UIKit

class ViewController: UIViewController{
    @IBOutlet weak var block: UIView!

    override func viewDidLoad(){
        super.viewDidLoad()


    }
    override func viewDidAppear(_ animated: Boo(){
        super.viewDidAppear(animated)
        DispatchQueue.main.asyncAfter(deadline: .now() + .seconds(2)){
            self.upMove()
        }
    }
```

func upMove(){ → by default ⇒ ease in, ease out
    UIView.animate(withDuration: 1, animations:{
        self.block.frame.origin = CGPoint(x: self.view.center.x-(self.block.frame.width/2), y:0)
    }){(success) in
        self.rightMove
    }
}

func rightMove(){
    -- — — — — — —
}

57-5 9:41

// timing function

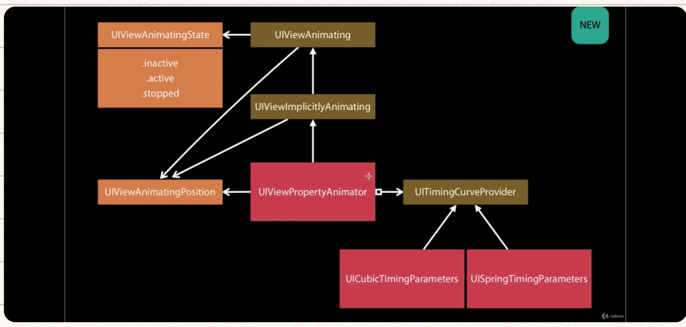
UIView.animate(withDuration: 2, delay: 2, options: .curveEaseIn, animations{ ----- }, completion{~~}

// spring

UIView.animate(withDuration: 2, delay: 2, usingSpringWithDamping: 0.5, initialSpringVelocity: 4
    options: .curveEaseIn, animations{ ----- }, completion{~~}

(0.5: 0~1 → over shoot most)

// Property Animation



// advantages:
- Dynamic Animations
- Interruptible
- Scrubbale
- Reveresible
- More Timing Functions

import UIKit
class ViewController: UIViewController{
    var animator: UIViewPropertyAnimator!

    @IBOutlet weak var block: UIView!
    override func viewDidLoad(){
        super.viewDidLoad()
        setupAnimation()
    }

    func setupAnimation(){
        self.block.transform = CGAffineTransform.identity ← make sure
        self.block.frame.origin.x = CGFloat(0)
        animator = UIViewPropertyAnimator(duration: 1.2, curve: UIView.AnimationCurve.linear, animations:{
            let screenWidth = self.view.frame.size.width
            let blockWidth = self.view.frame.size.width
            self.block.frame.origin.x = screenWidth - blockWidth
            self.block.transform = CGAffineTransform(rotationAngle: CGFloat.pi)
        })

```
        animator.addCompletion{ (position) in
            self.block.backgroundColor = UIColor.red
        }
    }
    @IBAction func startAnimation(_ sender: Any){
        if animator.isRunning{
            return
        }
        switch animator.state{
        case .active:
            animator.isReversed = true => will still call the completion handler
            animator.addAnimations{
                self.block.transform = CGAffineTransform(scaleX: 1.7, y: 1.7)
            }

            animator.startAnimation()
        case .inactive:
            setupAnimation()
            animator.startAnimation()
        case .stopped:
            animator.finishAnimation(at: UIViewAnimatingPosition.current)
            break
        }
    }

    @IBAction func stopAnimation(_ sender: Any){
        animator.stopAnimation(false)  // without finishing, if true => will become inactive
    }

    @IBAction func pauseAnimation(_ sender: Any){  <= ensure always active
        if animator.state == UIViewAnimatingState.active{
            animator.pauseAnimation()
        }
    }

    @IBAction func sliderDidChange(_ sender: Any){
        animator.fractionComplete = CGFloat(slider.value)
    }
```

```
// animate the login screen
import UIKit
class ViewController: UIViewController {
    lazy var logoImage: UIImageView = {
        let image = UIImage(named: "cat")!
        let imageView = UIImageView(image: image)
        imageView.layer.masksToBounds = true
        return imageView
    }()

    @IBOutlet weak var usernameTextField: UITextField!
    @IBOutlet weak var passwordTextField: UITextField!
    @IBOutlet weak var loginButton: UIButton!
    @IBOutlet weak var loginView: UIView!

    var moveLogoAnimator: UIViewPropertyAnimator!

    override func viewDidLoad() {
        super.viewDidLoad()                                   // important
        logoImage.translatesAutoresizingMaskIntoConstraints = false
        loginView.addSubview(logoImage)
        NSLayoutConstraint.activate([
            logoImage.centerXAnchor.constraint(equalTo: loginView.centerXAnchor),
            logoImage.centerYAnchor.constraint(equalTo: loginView.centerYAnchor),
            logoImage.heightAnchor.constraint(equalToConstant: CGFloat(100)),
            logoImage.widthAnchor.constraint(equalToConstant: CGFloat(100))
        ])
        loginView.transform = CGAffineTransform(scaleX: 0, y: 0)
        usernameTextField.alpha = CGFloat(0)
        passwordTextField.alpha = CGFloat(0)
        loginButton.alpha = CGFloat(0)
        loginView.backgroundColor = UIColor(red: 0.29, green: 0.29, blue: 0.29, alpha: 1.0)
    }

    override func viewDidLayoutSubviews() {
        super.viewDidLayoutSubviews()
        loginView.layer.cornerRadius = CGFloat(7.0)
        loginButton.layer.cornerRadius = CGFloat(5.0)
        logoImage.layer.cornerRadius = CGFloat(50.0)
    }

    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)                         // if no delay, sometimes we do not see the whole animation
        UIView.animate(withDuration: 0.8, delay: 1, usingSpringWithDamping: 0.5, initialSpringVelocity: 2, options: .curveEaseInOut,
            animations: {
                self.loginView.transform = CGAffineTransform(scaleX: 1.0, y: 1.0)
        }) { (success) in
            self.setupMoveLogoAnimation()
            self.moveLogoAnimator.startAnimation()
```

gray color ⇒ d8d8d8

space black ⇒ 4A4A4A

* make sure all views embeded properly

```
    }
}
```

-05:04

```
func setupMoveLogoAnimation(){
    moveLogoAnimator = UIViewPropertyAnimator(duration: 2.0, curve: .easeIn, animations: nil)

    moveLogoAnimator.addAnimations({
        self.logoImage.frame.origin.y = CGFloat(20.0)
        self.loginView.backgroundColor = UIColor.white
    }, delayFactor: 0.2)

    moveLogoAnimator.addAnimations({
        self.usernameTextField.alpha = 1.0
    }, delayFactor: 0.6)

    moveLogoAnimator.addAnimations({
        self.passwordTextField.alpha = 1.0
    }, delayFactor: 0.7)

    moveLogoAnimator.addAnimations({
        self.loginButton.alpha = 1.0
    }, delayFactor: 0.8)

}
```

// Constraint animations

```
import UIKit
class ViewController: UIViewController{
```