Every UIView comes with a default backing layer

Implicit and Explicit
↓ UIView animations
↓ CA Basic Animation
CA Key Frame Animation

Model and Presentation Layer tree

```
import UIKit

class ViewController : UIViewController{
    lazy var exampleView: UIView = {
        let view = UIView()
        view.backgroundColor = UIColor.orange
        view.frame = CGRect (x:0, y: self.view.center.y - 60, width:100, height:100)
        return view
    }()

    lazy var startButton : UIButton={
        let button = UIButton (frame: CGRect (x: self.view.center.x - 50, y: self.view.frame.height - 80, width:100, height:60))
        button.setTitle ("start", for: .normal)
        button.backgroundColor = UIColor.blue
        button.addTarget (self, action: #selector(startAnimating), for: .touchUpInside)
        return button
    }()

    override func viewDidLoad(){
        super.viewDidLoad()
        view.addSubview(exampleView)
        view.addSubview(startButton)
    }

    @objc func startAnimating(){
        let basicAnimation = CABasicAnimation(keyPath: "position.x")   //or #keyPath(CALayer.position)
        // exampleView.layer.position.x ↑same

        basicAnimation.fromValue = exampleView.frame.width/2   → the presentation layer got different origion
        //basicAnimation.byValue
        basicAnimation.toValue = 250.0
        basicAnimation.duration = 1.0
        exampleView.layer.setValue ("anything", forKey: "something")


        basicAnimation.beginTime = CACurrentMediaTime() + 1.0
        //basicAnimation.fillMode = CAMediaTimingFillMode.forwards
        // basicAnimation.isRemovedOnCompletion = false
        basicAnimation.delegate = self
        exampleView.layer.add(basicAnimation, forKey: "basic")
    }
}
```

```
extension ViewController : CAAnimationDelegate{
    func animationDidStart(_ anim : CAAnimation){    // update the location in the animationDidStart function
        example.layer.position.x = 250.0              // if there is a delay of the animation

    }

    func animationDidStop(_ anim : CAAnimation, finished flag: Bool){
        if flag{                                                    // works if "is Removed On Completion" = false
            // if anim == exampleView.layer.animation(forKey: "basic"){
            //     print("sucess")
            // }

            guard let theKey = exampleView.layer.value(forKey: "something") as? String else { return }
        }
    }
}

// Animated Gradient Layer
import UIKit

class ViewController : UIViewController{
    let gradient = CAGradientLayer()
    var gradientSet = [[CGColor]]()
    var currentGradient = 0                          // do not forget .cgColor
    let gradientOne = UIColor...    .cgColor
        // two, three

    override func viewDidLoad(){
        super.viewDidLoad()
        gradientSet.append([gradientOne, gradientTwo])
        gradientSet.append([gradientTwo, gradientThree])
        gradientSet.append([gradientThree, gradientOne])
        gradient.colors = gradientSet[currentGradient]
        gradient.startPoint = CGPoint(x:0, y:0)
        gradient.endPoint = CGPoint(x:1.0, y:1.0)
        gradient.drawsAsynchronously = true    // for continuous animes
        view.layer.addSublayer(gradient)
    }
    override func viewDidLayoutSubviews(){
        super.viewDidLayoutSubviews()
        gradient.frame = view.bounds
    }
    overview func viewDidAppear(_ animated: Bool){
        super.viewDidAppear(animated)
        animateGradient()
    }
```

```
    func animateGradient(){
        var previousGradient : Int!
        if currentGradient < gradientSet.count - 1{
            currentGradient += 1
            previousGradient = currentGradient - 1
        } else{
            currentGradient = 0
            previousGradient = gradientSet.count - 1
        }
        let gradientChangeAnim = CABasicAnimation(keyPath: "colors")
        gradientChangeAnim.duration = 5.0
        gradientChangeAnim.fromValue = gradientSet[previousGradient]
        gradientChangeAnim.toValue = gradientSet[currentGradient]
        gradient.setValue(currentGradient, forKeyPath: "ColorChanging")
        gradientChangeAnim.delegate = self
        gradient.add(gradientChangeAnim, forKey: nil)
    }
}

extension ViewController: CAAnimationDelegate{
    func animationDidStop(_ anim: CAAnimation, finished flag: Bool){
        if flag{
            if let colorChange = gradient.value(forKey: "colorChange") as? Int{
                gradient.colors = gradientSet[colorChange]
                animateGradient()                    // update the model layer
            }
        }
    }
}

// CAKeyFrameAnimation and shaking login Animation

import UIKit

class ViewController : UIViewController{
    @IBOutlet weak var passwordTextField: UITextField!
    let CORRECT_PASSWORD = "cats"
    override func viewDidLoad(){
        super.viewDidLoad()
        passwordTextField.layer.borderColor = UIColor.black.cgColor
        passwordTextField.layer.borderWidth = CGFloat(3.0)
        passwordTextField.layer.cornerRadius = CGFloat(5.0)
        passwordTextField.addTarget(self, action: #selector(passwordDidEnterText), for: UIControl.Event.editingDidEndOnExit)
        let viewIsTapped = UITapGestureRecognizer(target: self, action: #selector(viewIsTapped))
        view.addGestureRecognizer(viewIsTapped)
    }

    @objc func viewIsTapped(){
        view.endEditing(true)
```

```
}

@objc func passwordDidEnterText(){
    passwordTextField.text != CORRECT_PASSWORD{
        let shakeAnimation = CAKeyframeAnimation(keyPath: "position.x")
        shakeAnimation.values = [0, -15, 15, -15, 15, 0]
        shakeAnimation.keyTimes = [0, 0.2, 0.4, 0.6, 0.8, 1.0]
        shakeAnimation.duration = 0.4
        shakeAnimation.isAdditive = true  -> important
        passwordTextField.layer.add(shakeAnimation, forKey: nil)
    }
    else{
        print("password was correct")
    }

// standalone and backing layers
import UIKit

class ViewController: UIViewController{
    let standaloneLayer: CALayer = CALayer()
    let practiceView: UIView = UIView()

    enum selectableColors{
        case red, yellow, orange
        func getColor() -> CGColor{
            switch self{
                case .red:                    -> do not forget the cgColor
                    return UIColor.red.cgColor
                ↓ similar ones.
            }
        }
    }

var currentColor: SelectableColors = SelectableColors.red

override func viewDidLoad(){
    super.viewDidLoad()
    view.addSubview(practiceView)
    practiceView.frame = CGRect(x:0, y:0, width:100, height:100)
    practiceView.center = view.center
    standaloneLayer.backgroundColor = currentColor.getColor()
    practiceView.layer.addSublayer(standaloneLayer)

    let viewTap = UITapGestureRecognizer(target: self, action: #selector(viewWasTapped))
    practiceView.addGestureRecognizer(viewTap)
    practiceView.isUserInteractionEnabled = true
```

```
}

override func viewDidLayoutSubviews(){
    super.viewDidLayoutSubviews()
    standaloneLayer.frame = practiceView.bounds
}
```

// by default all layers have animation, UIView blocked all the backing layer's animations

```
@objc func viewWasTapped(){
    switch currentColor{
      case .red:
          currentColor = .orange
          standaloneLayer.backgroundColor = currentColor.getColor()
      ↓ similar other cases

    }
}
```

CABasicAnimation, CAKeyframeAnimation conforms to CAActionProtocol → render tree ↓ to model or presentation layer

// CATransaction

```
import UIKit

class ViewController: UIViewController{
    let practiceLayer: CALayer = CALayer()
    let practiceView: UIView = UIView()
    override func viewDidLoad(){
        super.viewDidLoad()
        view.addSubview(practiceView)
        practiceView.frame = CGRect(x:0, y:0, width:100, height:100)
        practiceView.layer.addSublayer(practiceLayer)
        practiceLayer.backgroundColor = UIColor.red.cgColor
        let viewTap = UITapGestureRecognizer(target: self, action: #selector(practiceDuration))
        practiceView.addGestureRecognizer(viewTap)
        practiceView.isUserInteractionEnabled = true
    }

    override func viewDidLayoutSubviews(){
        super.viewDidLayoutSubviews()
        practiceLayer.frame = practiceView.bounds
    }

    @objc func practiceDuration(){
        CATransaction.begin()
        CATransaction.setAnimationDuration(2)
        practiceLayer.position = CGPoint(x: 300, y: 300)
        practiceLayer.backgroundColor = UIColor.orange.cgColor
        CATransaction.commit()
    }
```

// CATransaction.begin()
// ⋮ some code change properties
// CATransaction.commit()

```
@objc func practiceTimingFunctionAnime(){
    let timingFunction = CAMediaTimingFunction(name: CAMediaTimingFunctionName.easeIn)
    CATransaction.begin()
    CATransaction.setAnimationTimingFunction(timingFunction)
    practiceLayer.position = CGPoint(x: 300, y: 300)
    practiceLayer.backgroundColor = UIColor.orange.cgColor

    CATransaction.commit()
}

@objc func practiceDisableAnim(){
    CATransaction.begin()
    CATransaction.setDisableActions(true)  // no animation will happen
    UIView.animate(withDuration: 1.0){
        self.practiceView.alpha = 0.0
    }
    CATransaction.commit()
}

@objc func practiceAnimComplete(){
    CATransaction.begin()
    CATransaction.setCompletionBlock{
        self.practiceView.layer.position.x = 250

    }
    UIView.animate(withDuration: 1.0){
        self.practiceView.alpha = 1.0
    }
    let basicAnimation = CABasicAnimation(keyPath: "position.x")
    basicAnimation.duration = 2.0
    basicAnimation.fromValue = practiceView.layer.position
    basicAnimation.toValue = 250
    practiceView.layer.add(basicAnimation, forKey: nil)
    CATransaction.commit()
}

}
```

// Grouping Animations

```
import UIKit

class ViewController: UIViewController{

    lazy var practiceView: UIView = {
        let _practiceView = UIView()
        _practiceView.frame = CGRect(x:0, y:30, width:100, height:100)
        _practiceView.layer.backgroundColor = UIColor.red.cgColor
        return _practiceView
    }()

    override func viewDidLoad(){
        super.viewDidLoad()
        view.addSubview(practiceView)
        let viewTap = UITapGestureRecognizer(target: self, action: #selector(animateGroup))
        practiceView.addGestureRecognizer(viewTap)
        practiceView.isUserInteractionEnabled = true
    }

    @objc func animateGroup(){
        let basicAnimation = CABasicAnimation(keyPath: "position.x")
        basicAnimation.fromValue = practiceView.layer.position.x
        basicAnimation.toValue = practiceView.layer.position.x + 250
        let keyFrameAnimation = CAKeyFrameAnimation(keyPath: "backgroundColor")
        keyFrameAnimation.values = [UIColor.red.cgColor, ....]
        keyFrameAnimation.keyTimes = [0, 0.25, 0.75, 1]
        let animGroup = CAAnimationGroup()
        animGroup.duration = 4
        animGroup.animations = [basicAnimation, keyFrameAnimation]
        practiceView.layer.add(animGroup, forKey: nil)
    }
```