IOS MIT SWIFT Workshop

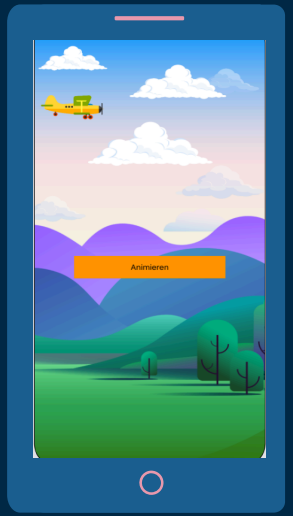
UIVIEWPROPERTYANIMATOR

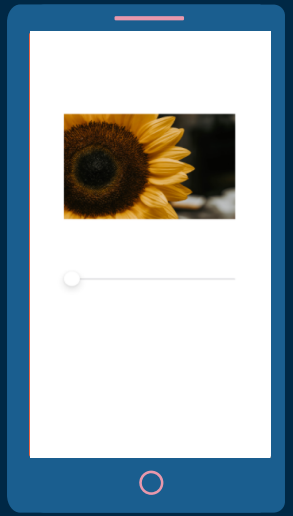
Lisa Haas

Ablauf

- Theoretische Einführung
- Live Beispiele
- Übung

SNEAK PEEK





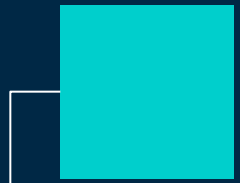
https://github.com/lisajhaas/fsios-workshop

UIViewPropertyAnimator

- Eine Klasse, welche Änderungen bei Views (Propertys) animiert und die dynamische Modifizierung dieser Animationen erlaubt.

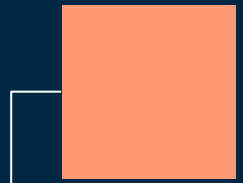
- Animationen können vom Start bis zum Ende normal oder interaktiv durchgeführt werden.

Diese UI View Propertys können geändert werden



01

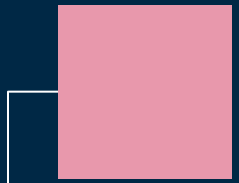
frame/center



02

bounds

vorübergehende Änderung der Größe



03

transform

translation, rotieren, skalieren

04

alpha

Transparenz

05

backgroundColor

UIViewPropertyAnimator

1. Animator Objekt erstellen UIViewPropertyAnimtor()
2. Animationen hinzufügen .addAnimations() - optional
3. Completion Handler hinzufügen .addCompletion() - optional
4. Animation starten .startAnimation()

Initialisierung

```
let animate = UIViewPropertyAnimator(
    duration: TimeInterval,
    curve: UIView.AnimationCurve,
    animations: (() -> Void)?)
```

Dauer der Animation

Zeitliche Kurve der Animation

Änderung der Propertys

Curve Möglichkeiten: .linear, .easeIn, .easeInOut, .easeOut

Vergleich

UIView.animate

- statisch

UIViewPropertyAnimator

- Interaktiv
- Unterbrechbar
- Dynamisch anpassbar (Completion Handler hinzufügen usw.)

Code Beispiel

```
UIViewPropertyAnimator.runningPropertyAnimator(
    withDuration: TimeInterval,
    delay: TimeInterval,
    options: UIView.AnimationOptions,
    animations: () -> Void,
    completion: ((UIViewAnimatingPosition) -> Void)?)
```

Dauer der Animation

Dauer, bevor die Animation startet

Optionen, wie Animation ausgeführt werden soll

Änderung der Propertys

Wird am Ende der Animation ausgeführt

Ähnlich wie UIView.animate – Es können allerdings z.B. weitere Animationen und Completions hinzugefügt werden: .addAnimations() .addCompletion() (Funktionen von UIViewPropertyAnimator

Code Beispiel

```
if myView.alpha == 1.0 {
  UIViewPropertyAnimator.runningPropertyAnimator(
      withDuration: 3.0,
      delay: 2.0,
      options: [.allowUserInteraction],
      animations: {myView.alpha = 0.0 },
      completion: {if $0 == .end {myView.removeFromSuperview()}})

  }
```

UIVIEWANIMATION OPTIONS

beginFromCurrentState

Nimmt die eigentlichen Property Einstellungen auf.

repeat

Auf bestimmte Zeit wiederholen

allowUserInteraction

Nutzereingaben werden während der Animation erlaubt.

autoreverse

Animationen vorwärts und dann rückwärts abspielen

layoutSubviews

Animiert das Relayout einer Subview mit einer Eltern-Animation

overrideInheritedCurve

Wenn nicht gesetzt, dann soll die Dauer einer aktuell ablaufenden Animation verwendet werden.

UIVIEWANIMATION OPTIONS

overrideInheritedCurve

Wenn nicht gesetzt, dann soll curve einer aktuell ablaufenden Animation verwendet werden

allowAnimatedContent

Wenn nicht gesetzt, interpoliere zwischen aktuellen und end „bits“

curveEaseIn

Langsamer am Anfang und dann bis zum Schluss konsistent

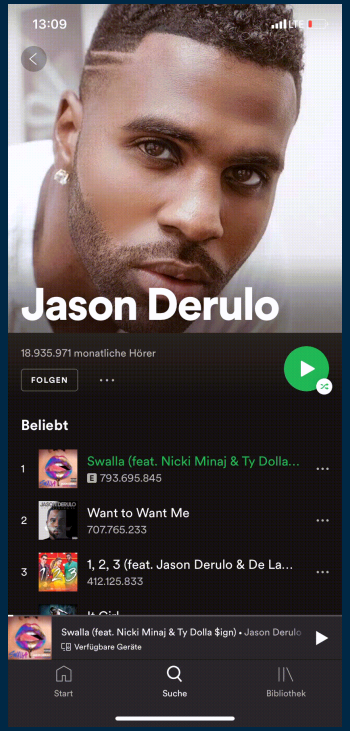
curveEaseInEaseOut

Langsam am Anfang, dann normal und am Ende wieder langsam

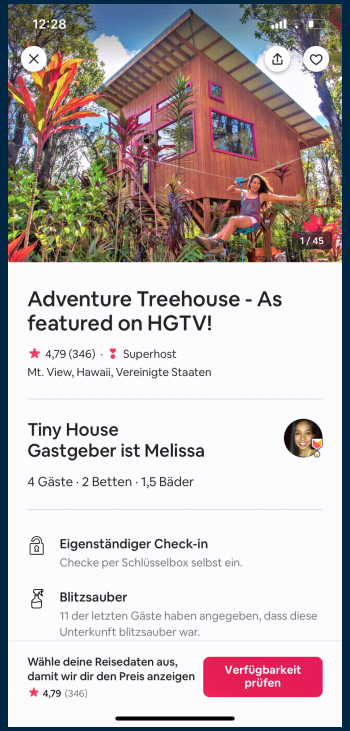
curveLinear

Durchgehend dieselbe Geschwindigkeit

APP BEISPIELE



APP BEISPIELE



APP BEISPIELE

LIVE BEISPIELE

https://github.com/lisajhaas/fsios-workshop

Gibt es noch Fragen?

DANKE

QUELLEN

IOS DEVELOPER DOCUMENTATION

https://developer.apple.com/documentation/uikit/uiviewpropertyanimator

STANFORD KURS ANIMATIONS

https://www.youtube.com/watch?v=wERNQyfJYLo&t=131s