Full Stack iOS Entwicklung mit Swift

WPF FSIOS

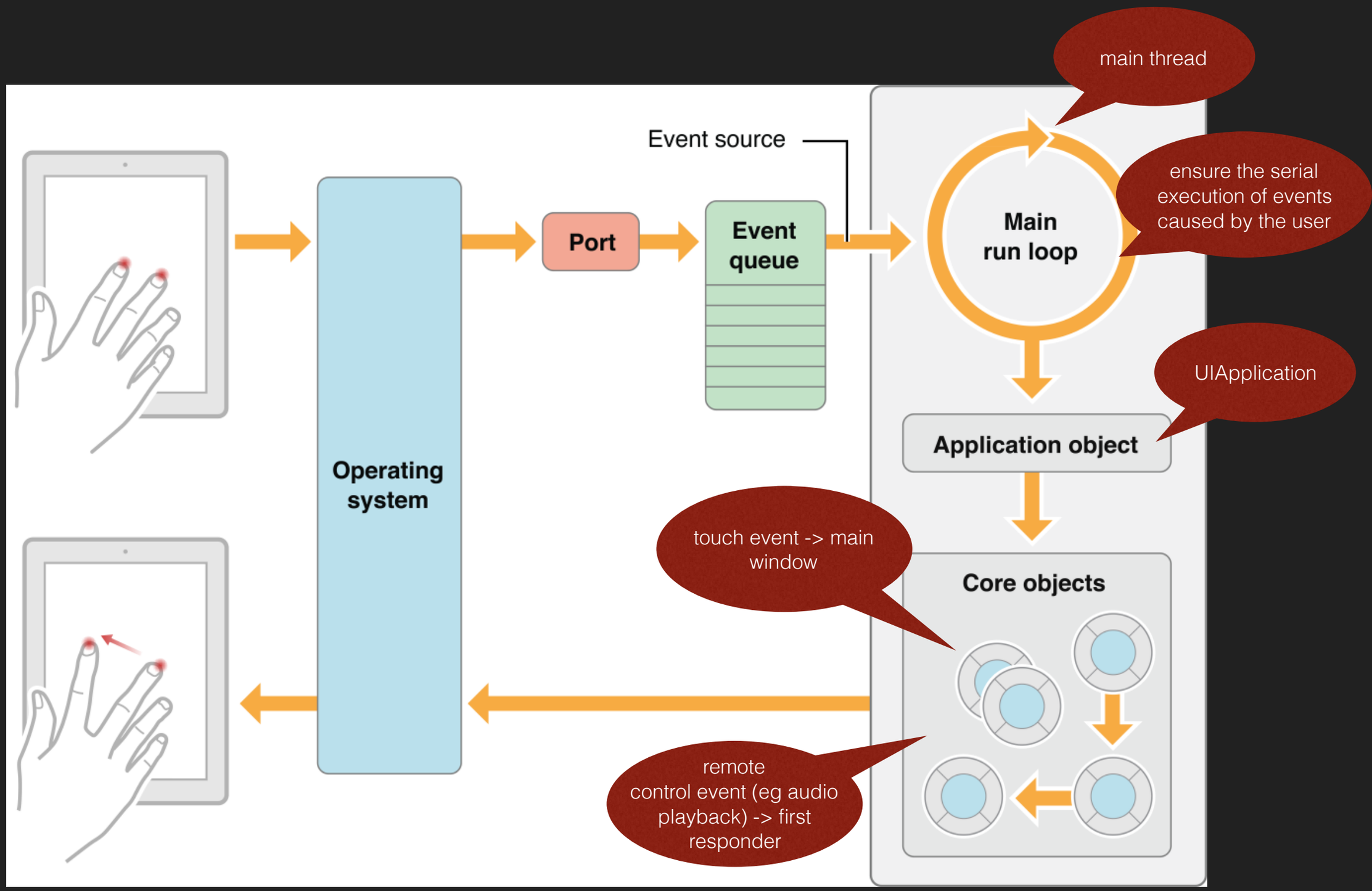
Alexander Dobrynin, M.Sc.

Heute

Application Lifecycle
ViewController Lifecycle
Segues

Zusammenfassung

Application Lifecycle



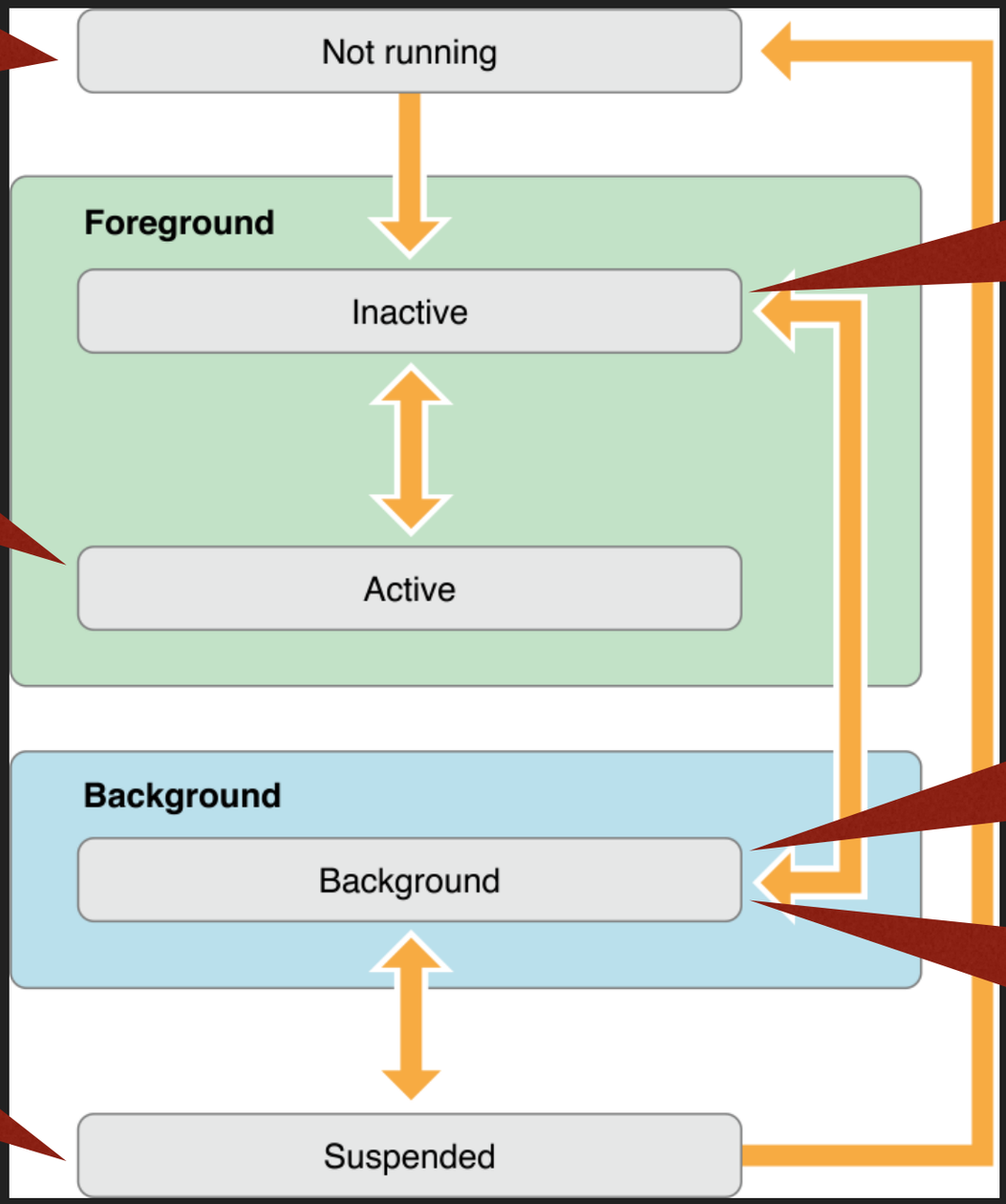
https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html

Application Lifecycle

- UIApplication koordiniert die Interaktion zwischen dem System (iOS) und der eigenen App
- Die Klasse AppDelegate ist der Delegate (Reagierende) von UIApplication und der Einstiegspunkt der eigenen App
- Als Delegate findet hier die Reaktion auf Application Lifecycle Events statt
- Typische Events sind
  - App wird gestartet
  - App wird gekillt
  - App bekommt Zeit für Hintergrundaktualisierungen
- Die Events können auch über das Notification-Pattern unter dem jeweiligen Notification.Name.* abonniert werden
- Achtung: Es gilt als Anti-Pattern, wenn man State (z.B. Einstellungen) und Anwendungslogik in der AppDelegate hat. Es soll lediglich ein Delegate für Events sein

Application Lifecycle

app has not been launched or was running but was terminated by the system



app is running in the foreground but is currently not receiving events

app is running in the foreground and is receiving events. This is the normal mode for foreground apps.

app is in the background and executing code. Most apps enter this state briefly on their way to being suspended …

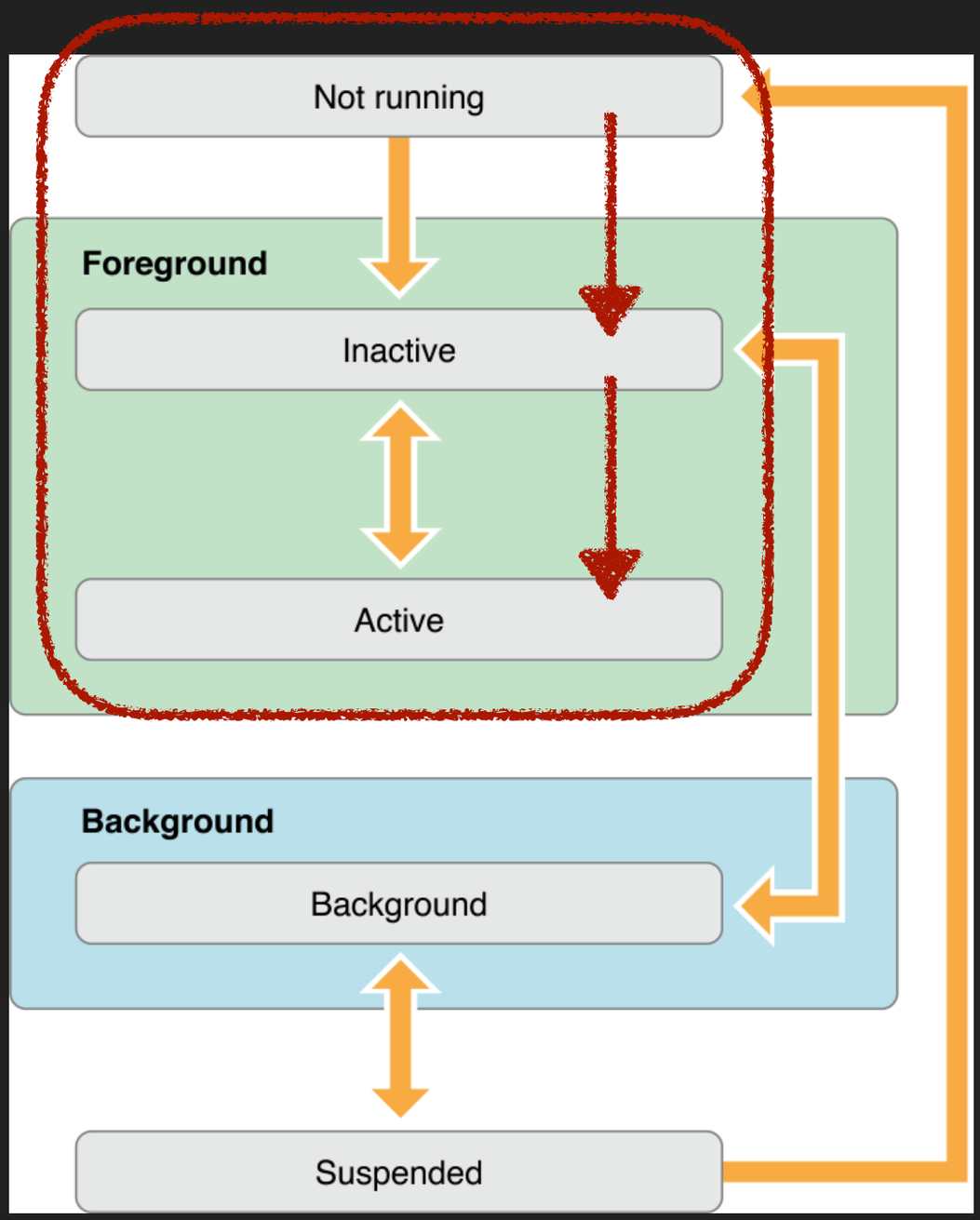
app is in the background but is not executing code. While suspended, an app remains in memory […]

… However, an app that requests extra execution time may remain in this state for a period of time

https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html

Application Lifecycle

Launching App



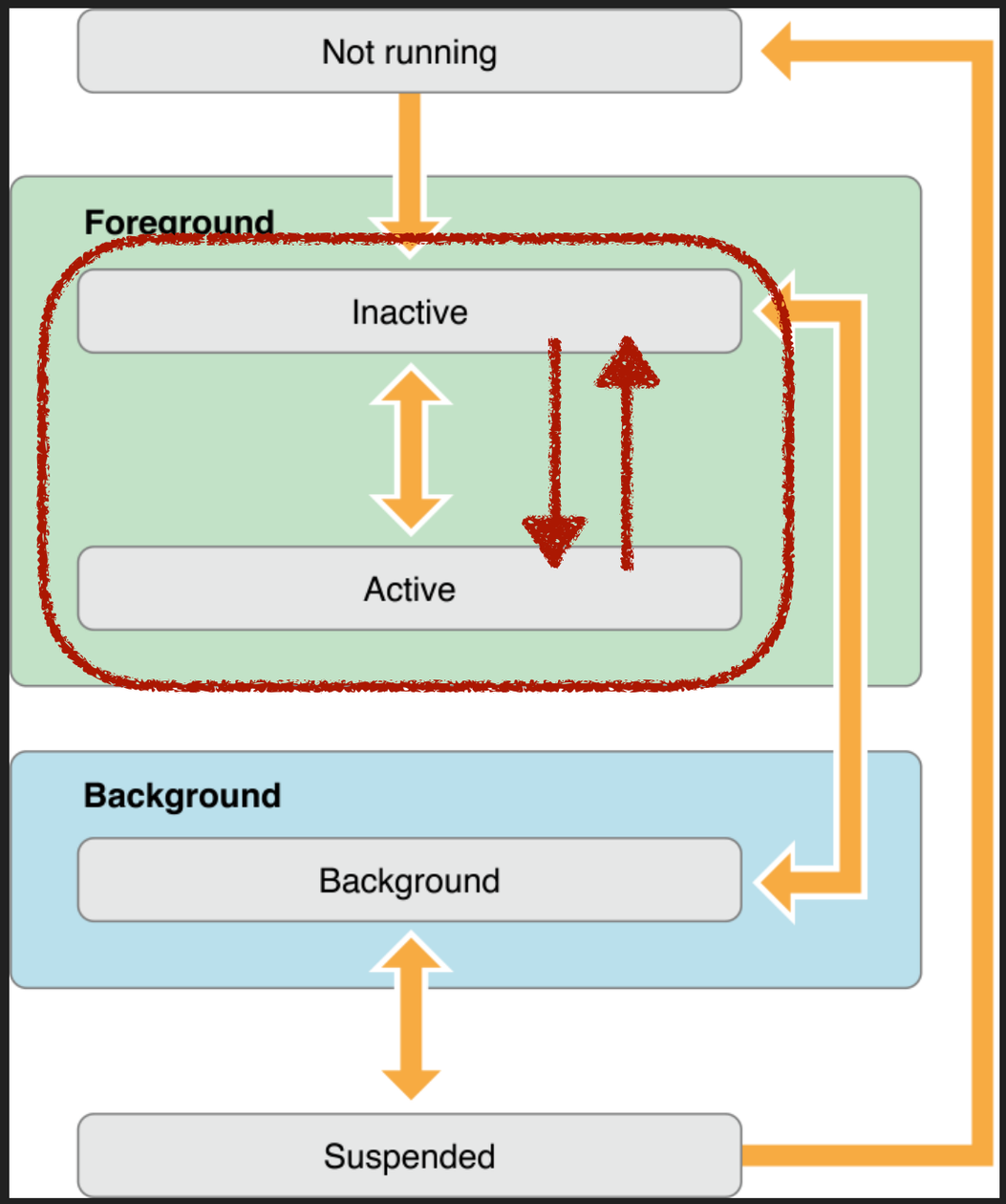
https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html

Application Lifecycle

Double Home Button
Pull Control Center
Pull Notification Center

…

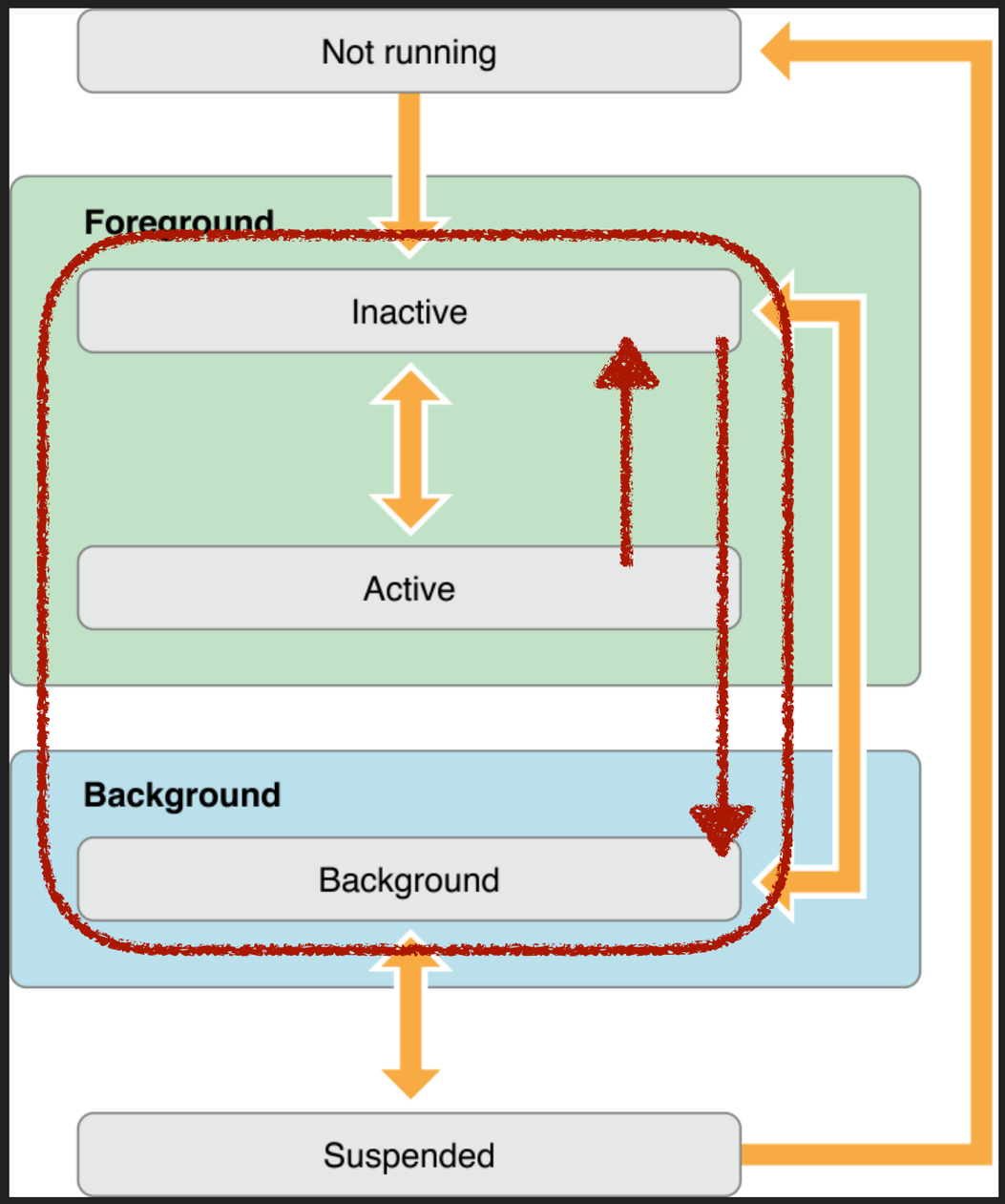
Undo



https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html

Application Lifecycle

Single Home Button



https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html

Application Lifecycle

```swift
@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationDidBecomeActive(_ application: UIApplication) {
        // Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the background, optionally refresh the user interface.
    }

    func applicationWillResignActive(_ application: UIApplication) {
        // Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.
        // Use this method to pause ongoing tasks, disable timers, and invalidate graphics rendering callbacks. Games should use this method to pause the game.
    }

    func applicationDidEnterBackground(_ application: UIApplication) {
        // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.
        // If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.
    }

    func applicationWillEnterForeground(_ application: UIApplication) {
        // Called as part of the transition from the background to the active state; here you can undo many of the changes made on entering the background.
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:.
    }
}
```

Heute

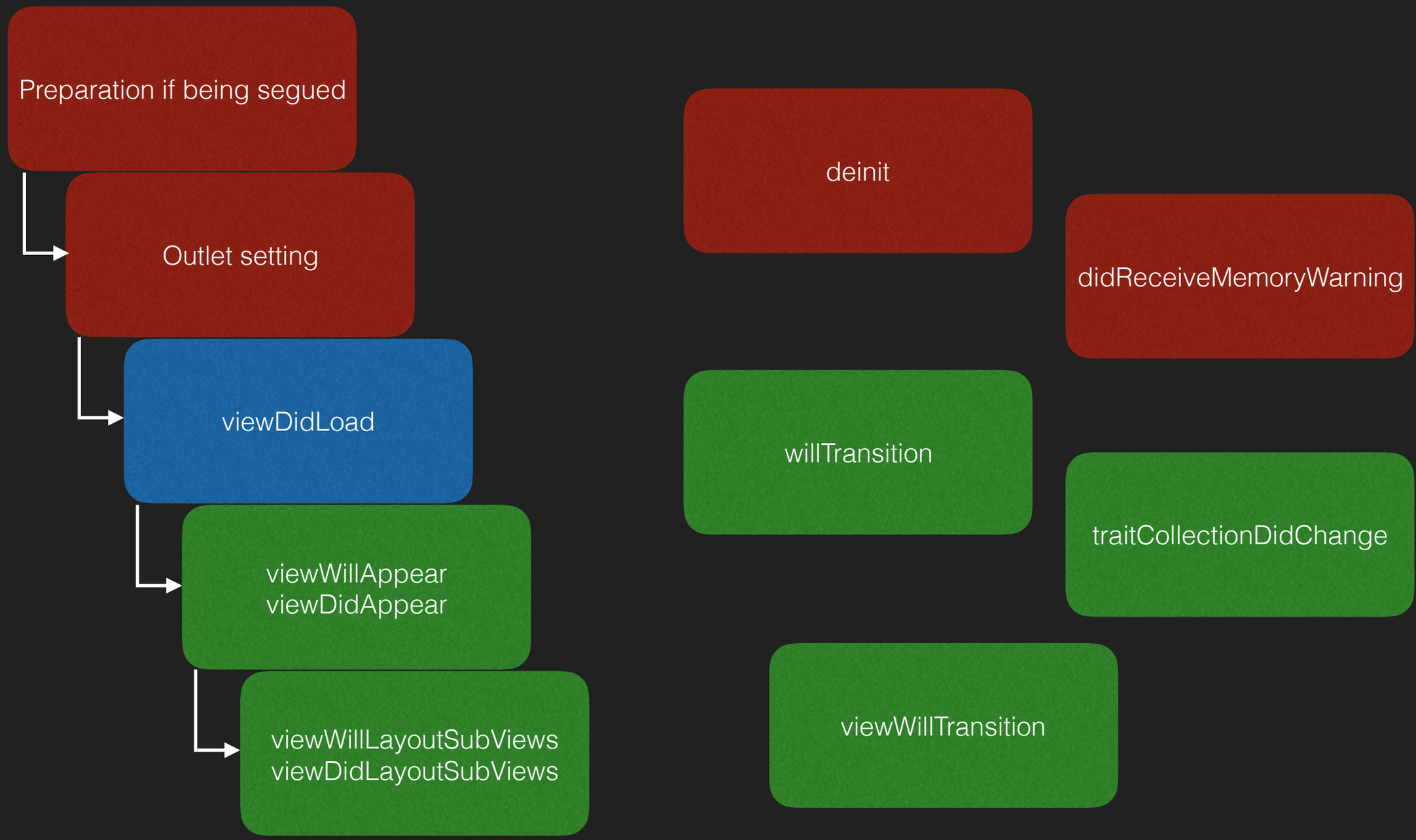
Application Lifecycle
ViewController Lifecycle
Segues

Zusammenfassung

ViewController Lifecycle

- Ähnlich zum Application Lifecycle, durchlebt jeder ViewController ebenfalls einen definierten Lifecycle
- Typische Events sind
    - ViewController ist vollständig instanziiert
    - ViewController ist/wird sichtbar und/oder unsichtbar
    - ViewController wurde rotiert
- Zudem sind das Setzten von Outlets und das Vorbereiten eines ViewControllers (Transitionen zwischen VC's, Segues) ebenfalls ein Teil des Lifecycles
- Achtung: Das Vorbereiten eines ViewController ist das Erste was passiert, weshalb man nicht auf Outlets, andere Properties oder Funktionen zugreifen kann

ViewController Lifecycle



ViewController Lifecycle

```swift
class LifecycleViewController : UIViewController {

    var model: Model? // 1. set in preparation if being segued

    var label: UILabel? // 2. outlets, set by the storyboard
    var imageView: UIImageView?
    var spinner: UIActivityIndicatorView?

    override func viewDidLoad() { // 3. preparation is finished and outlets are set. best place for init code
        super.viewDidLoad()

        label?.textColor = .white
        imageView?.contentMode = .scaleAspectFill
        spinner?.hidesWhenStopped = true
    }

    override func viewWillAppear(_ animated: Bool) { // 4. vc is about to be displayed on screen. show user content. 'viewDidAppear' is called after
        super.viewWillAppear(animated)
        guard let model = model else { return }

        label?.text = model.message
        spinner?.startAnimating()

        downloadImage(by: model.imageUrl) { image in
            self.imageView?.image = image
            self.spinner?.stopAnimating()
        }
    }

    override func viewWillTransition(to size: CGSize, with coordinator: UIViewControllerTransitionCoordinator) { // At some point the device is rotated
        super.viewWillTransition(to: size, with: coordinator)
        imageView?.contentMode = size.width > size.height ? .scaleAspectFit : .scaleAspectFill
    }

    override func viewWillDisappear(_ animated: Bool) { // At some point the vc will be removed from screen. clean up or undo `willViewAppear`. `viewDidDisappear` is called after
        super.viewWillDisappear(animated)
        spinner?.stopAnimating() // just in case
    }
}
```

ViewController Lifecycle

```swift
class LifecycleViewController : UIViewController {

    var model: Model? // 1. set in preparation if being segued

    var label: UILabel? // 2. outlets, set by the storyboard
    var imageView: UIImageView?
    var spinner: UIActivityIndicatorView?

    override func viewDidLoad() { // 3. preparation is finished and outlets are set. best place for init code
        super.viewDidLoad()

        label?.textColor = .white
        imageView?.contentMode = .scaleAspectFill
        spinner?.hidesWhenStopped = true
    }

    override func viewWillAppear(_ animated: Bool) { // 4. vc is about to be displayed on screen. show user content. 'viewDidAppear' is called after
        super.viewWillAppear(animated)
        guard let model = model else { return }

        label?.text = model.message
        spinner?.startAnimating()

        downloadImage(by: model.imageUrl) { image in
            self.imageView?.image = image
            self.spinner?.stopAnimating()
        }
    }

    override func viewWillTransition(to size: CGSize, with coordinator: UIViewControllerTransitionCoordinator) { // At some point the device is rotated
        super.viewWillTransition(to: size, with: coordinator)
        imageView?.contentMode = size.width > size.height ? .scaleAspectFit : .scaleAspectFill
    }

    override func viewWillDisappear(_ animated: Bool) { // At some point the vc will be removed from screen. clean up or undo `willViewAppear`. `viewDidDisappear` is called after
        super.viewWillDisappear(animated)
        spinner?.stopAnimating() // just in case
    }
}
```

ViewController Lifecycle

```swift
class LifecycleViewController : UIViewController {

    var model: Model? // 1. set in preparation if being segued

    var label: UILabel? // 2. outlets, set by the storyboard
    var imageView: UIImageView?
    var spinner: UIActivityIndicatorView?

    override func viewDidLoad() { // 3. preparation is finished and outlets are set. best place for init code
        super.viewDidLoad()

        label?.textColor = .white
        imageView?.contentMode = .scaleAspectFill
        spinner?.hidesWhenStopped = true
    }

    override func viewWillAppear(_ animated: Bool) { // 4. vc is about to be displayed on screen. show user content. 'viewDidAppear' is called after
        super.viewWillAppear(animated)
        guard let model = model else { return }

        label?.text = model.message
        spinner?.startAnimating()

        downloadImage(by: model.imageUrl) { image in
            self.imageView?.image = image
            self.spinner?.stopAnimating()
        }
    }

    override func viewWillTransition(to size: CGSize, with coordinator: UIViewControllerTransitionCoordinator) { // At some point the device is rotated
        super.viewWillTransition(to: size, with: coordinator)
        imageView?.contentMode = size.width > size.height ? .scaleAspectFit : .scaleAspectFill
    }

    override func viewWillDisappear(_ animated: Bool) { // At some point the vc will be removed from screen. clean up or undo `willViewAppear`. `viewDidDisappear` is called after
        super.viewWillDisappear(animated)
        spinner?.stopAnimating() // just in case
    }
}
```

ViewController Lifecycle

```
class LifecycleViewController : UIViewController {

    var model: Model? // 1. set in preparation if being segued

    var label: UILabel? // 2. outlets, set by the storyboard
    var imageView: UIImageView?
    var spinner: UIActivityIndicatorView?

    override func viewDidLoad() { // 3. preparation is finished and outlets are set. best place for init code
        super.viewDidLoad()

        label?.textColor = .white
        imageView?.contentMode = .scaleAspectFill
        spinner?.hidesWhenStopped = true
    }

    override func viewWillAppear(_ animated: Bool) { // 4. vc is about to be displayed on screen. show user content. 'viewDidAppear' is called after
        super.viewWillAppear(animated)
        guard let model = model else { return }

        label?.text = model.message
        spinner?.startAnimating()

        downloadImage(by: model.imageUrl) { image in
            self.imageView?.image = image
            self.spinner?.stopAnimating()
        }
    }

    override func viewWillTransition(to size: CGSize, with coordinator: UIViewControllerTransitionCoordinator) { // At some point the device is rotated
        super.viewWillTransition(to: size, with: coordinator)
        imageView?.contentMode = size.width > size.height ? .scaleAspectFit : .scaleAspectFill
    }

    override func viewWillDisappear(_ animated: Bool) { // At some point the vc will be removed from screen. clean up or undo `willViewAppear`. `viewDidDisappear` is called after
        super.viewWillDisappear(animated)
        spinner?.stopAnimating() // just in case
    }
}
```

ViewController Lifecycle

```swift
class LifecycleViewController : UIViewController {

    var model: Model? // 1. set in preparation if being segued

    var label: UILabel? // 2. outlets, set by the storyboard
    var imageView: UIImageView?
    var spinner: UIActivityIndicatorView?

    override func viewDidLoad() { // 3. preparation is finished and outlets are set. best place for init code
        super.viewDidLoad()

        label?.textColor = .white
        imageView?.contentMode = .scaleAspectFill
        spinner?.hidesWhenStopped = true
    }

    override func viewWillAppear(_ animated: Bool) { // 4. vc is about to be displayed on screen. show user content. 'viewDidAppear' is called after
        super.viewWillAppear(animated)
        guard let model = model else { return }

        label?.text = model.message
        spinner?.startAnimating()

        downloadImage(by: model.imageUrl) { image in
            self.imageView?.image = image
            self.spinner?.stopAnimating()
        }
    }

    override func viewWillTransition(to size: CGSize, with coordinator: UIViewControllerTransitionCoordinator) { // At some point the device is rotated
        super.viewWillTransition(to: size, with: coordinator)
        imageView?.contentMode = size.width > size.height ? .scaleAspectFit : .scaleAspectFill
    }

    override func viewWillDisappear(_ animated: Bool) { // At some point the vc will be removed from screen. clean up or undo `willViewAppear`. `viewDidDisappear` is called after
        super.viewWillDisappear(animated)
        spinner?.stopAnimating() // just in case
    }
}
```

ViewController Lifecycle

```swift
class LifecycleViewController : UIViewController {

    var model: Model? // 1. set in preparation if being segued

    var label: UILabel? // 2. outlets, set by the storyboard
    var imageView: UIImageView?
    var spinner: UIActivityIndicatorView?

    override func viewDidLoad() { // 3. preparation is finished and outlets are set. best place for init code
        super.viewDidLoad()

        label?.textColor = .white
        imageView?.contentMode = .scaleAspectFill
        spinner?.hidesWhenStopped = true
    }

    override func viewWillAppear(_ animated: Bool) { // 4. vc is about to be displayed on screen. show user content. 'viewDidAppear' is called after
        super.viewWillAppear(animated)
        guard let model = model else { return }

        label?.text = model.message
        spinner?.startAnimating()

        downloadImage(by: model.imageUrl) { image in
            self.imageView?.image = image
            self.spinner?.stopAnimating()
        }
    }

    override func viewWillTransition(to size: CGSize, with coordinator: UIViewControllerTransitionCoordinator) { // At some point the device is rotated
        super.viewWillTransition(to: size, with: coordinator)
        imageView?.contentMode = size.width > size.height ? .scaleAspectFit : .scaleAspectFill
    }

    override func viewWillDisappear(_ animated: Bool) { // At some point the vc will be removed from screen. clean up or undo `willViewAppear`. `viewDidDisappear` is called after
        super.viewWillDisappear(animated)
        spinner?.stopAnimating() // just in case
    }
}
```

ViewController Lifecycle

```
class BadLifecycleViewController : UIViewController {

    var model: Model? {
        didSet {
            label.text = model!.message // will crash, because outlets are implicitly unwrapped (which is done by default when using storyboards)
        }
    }

    @IBOutlet weak var label: UILabel! {
        didSet {
            label.textColor = .white // this should be fine, because `label` is just set
        }
    }

    @IBOutlet weak var imageView: UIImageView! {
        didSet {
            imageView.contentMode = .scaleAspectFill
        }
    }

    @IBOutlet weak var spinner: UIActivityIndicatorView! {
        didSet {
            spinner.hidesWhenStopped = true
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        // nothing to do here
    }
}
```

ViewController Lifecycle

```
class BadLifecycleViewController : UIViewController {

    var model: Model? {
        didSet {
            label.text = model!.message // will crash, because outlets are implicitly unwrapped (which is done by default when using storyboards)
        }
    }

    @IBOutlet weak var label: UILabel! {
        didSet {
            label.textColor = .white // this should be fine, because `label` is just set
        }
    }

    @IBOutlet weak var imageView: UIImageView! {
        didSet {
            imageView.contentMode = .scaleAspectFill
        }
    }

    @IBOutlet weak var spinner: UIActivityIndicatorView! {
        didSet {
            spinner.hidesWhenStopped = true
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        // nothing to do here
    }
}
```

ViewController Lifecycle

```
class BadLifecycleViewController : UIViewController {

    var model: Model? {
        didSet {
            label.text = model!.message // will crash, because outlets are implicitly unwrapped (which is done by default when using storyboards)
        }
    }

    @IBOutlet weak var label: UILabel! {
        didSet {
            label.textColor = .white // this should be fine, because `label` is just set
        }
    }

    @IBOutlet weak var imageView: UIImageView! {
        didSet {
            imageView.contentMode = .scaleAspectFill
        }
    }

    @IBOutlet weak var spinner: UIActivityIndicatorView! {
        didSet {
            spinner.hidesWhenStopped = true
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        // nothing to do here
    }
}
```

Heute

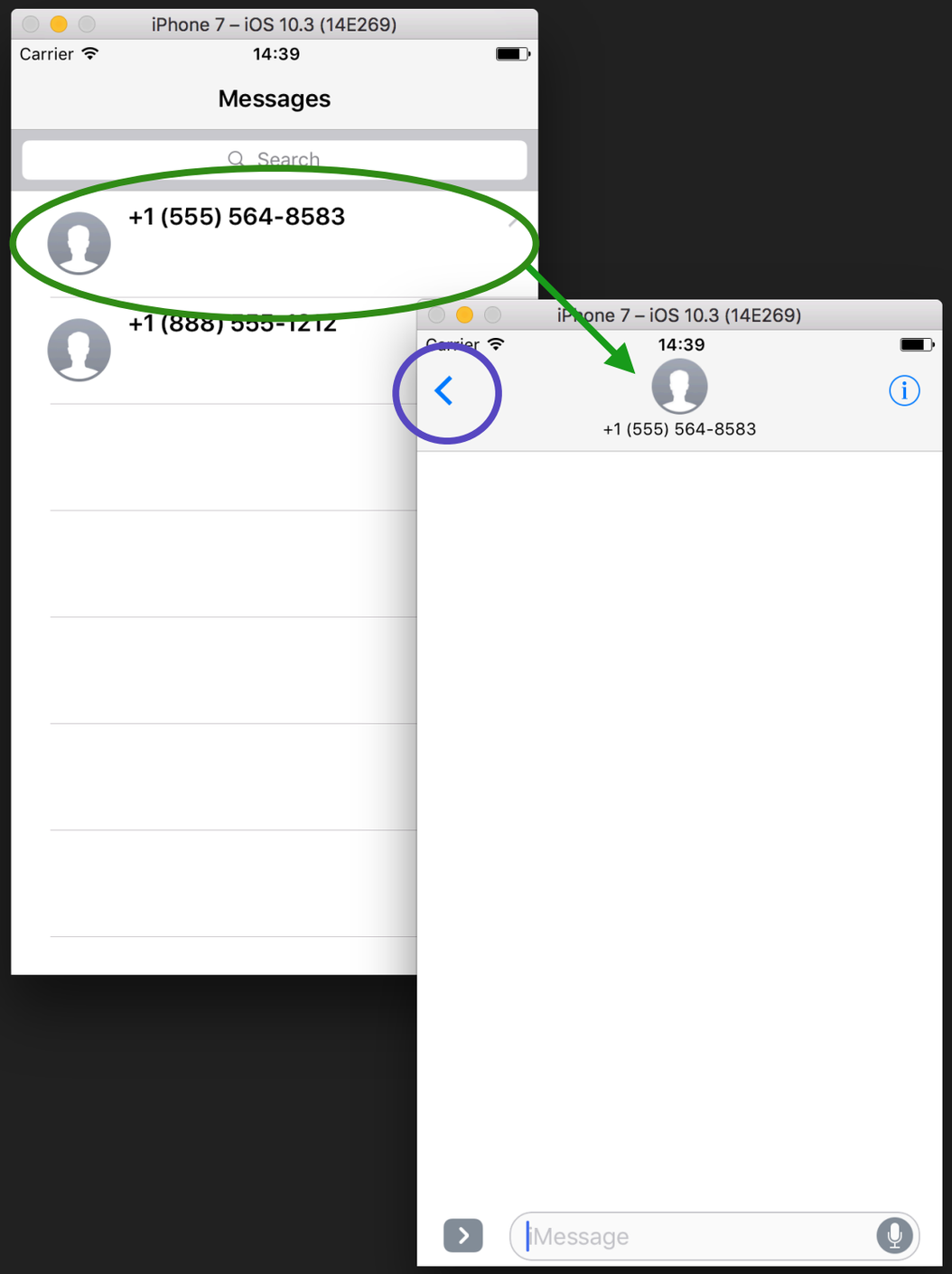
Application Lifecycle
ViewController Lifecycle
Segues

Zusammenfassung

Segues

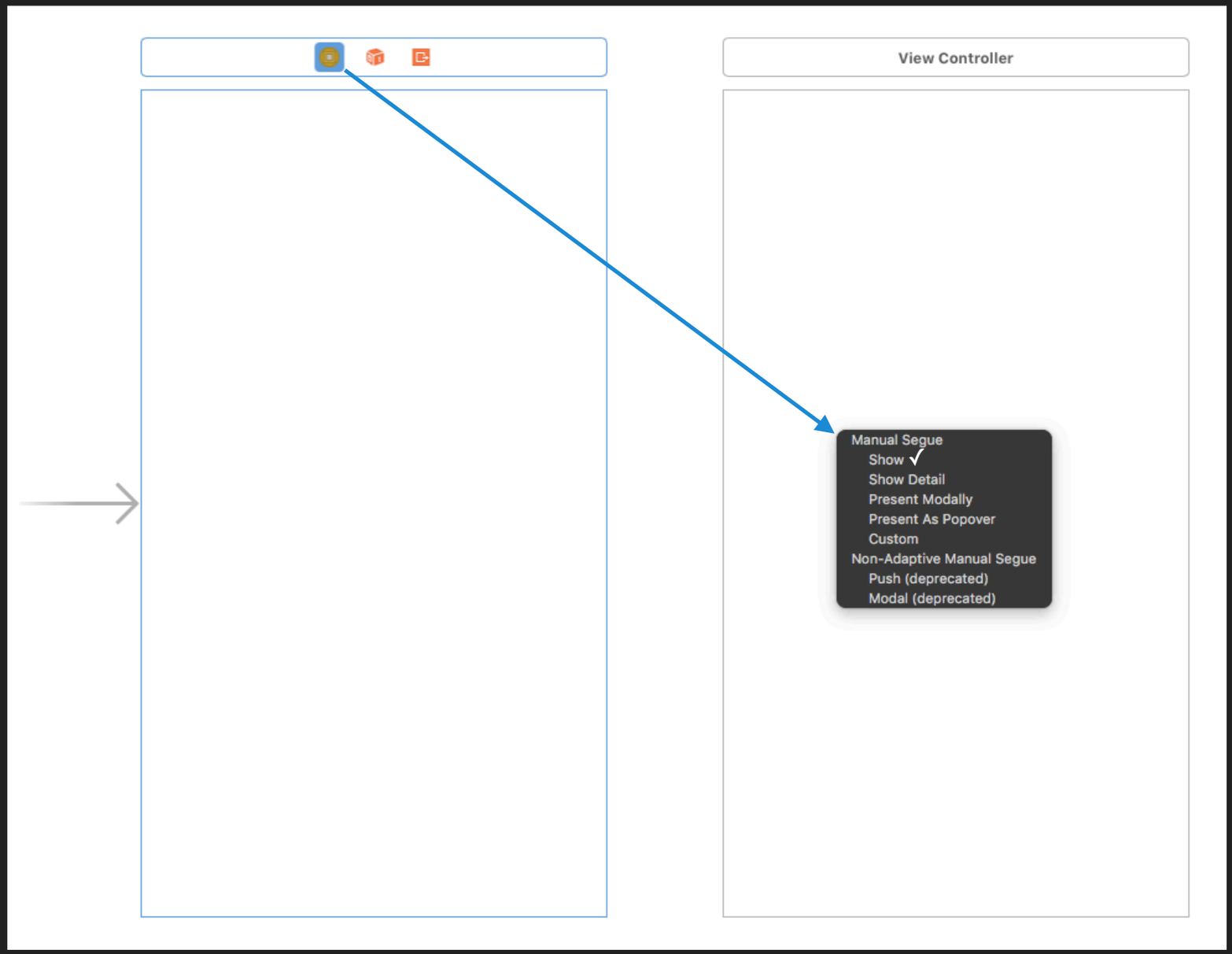
- Segues sind Transitionen zwischen zwei ViewController Src und Dest, wobei Src den kompletten MVC-Stack von Dest präsentieren möchte
- iOS kennt fünf Arten von Segues: Detail, Modal, Popover, Unwind und Embed
    - Detail-Segues pushen den Dest-VC auf den aktuellen Navigations-Stack
    - Modal-und Popover Segues präsentieren den Dest-VC in einem UIModalPresentationStyle (.fullScreen, .overFullScreen, .popover) und animiert nach einem UIModalTransitionStyle (.coverVertical, .flipHorizontal)
    - Unwind-Segues kehren von einem Dest-VC zum Src-VC zurück (eine Möglichkeit den Dest-VC zu verlassen)
    - Embed-Segues betteten einen vollständigen Dest-VC in den Src-VC innerhalb einer ContainerView ein
- Segues können im Storyboard entweder mit einem Ctrl-Drag zwischen Src-View und Dest-VC (Segue wird bei Interaktion mit View direkt ausgelöst) oder Src-VC und Dest-VC (Segue wird manuell ausgelöst) modelliert werden
- Der Src-VC kann die Segue vorbereiten, um bswp. das Model des Dest-VC zu setzten oder sonstigen Payload zu übertragen
- Achtung: Beim Vorbereiten ist auf den VC-Lifecycle zu achten. Die Outlets des Dest-VC werden erst nach der Preparation gesetzt
- Ein Dest-VC muss nicht unbedingt mittels einer Segue präsentiert werden. Transitionen zwischen ViewController können vollständig im Code implementiert werden. Segues sind der empfohlene Weg von Apple

Segues



- Detail: Erstellt und präsentiert einen MVC, indem der VC auf den aktuellen Navigations-Stack gepushed wird
- Modal: Erstellt und präsentiert ein MVC über den gesamten Screen
- Popover: Fast das gleiche wie Modal, nur dass es beim iPad “über” den Screen gelegt wird
- Unwind: Erstellt kein neues MVC, sondern kehrt zu einem bestehenden MVC zurück
- Embed: Einen gesamten MVC-Stack innerhalb eines VC “einbetten”

Segues


View Controller
Manual Segue
Show ✓
Show Detail
Present Modally
Present As Popover
Custom
Non-Adaptive Manual Segue
Push (deprecated)
Modal (deprecated)

Segues

```swift
class DetailSegueViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.leftBarButtonItem = UIBarButtonItem(title: "Show Detail", style: .plain, target: self, action: #selector(showDetail(_:)))
    }

    @objc func showDetail(_ barButton: UIBarButtonItem) {
        showBySegue(barButton) // show detail vc by using segues
        showByNavigationController(barButton) // show detail vc by pushing onto navigationController stack
    }

    private func showBySegue(_ barButton: UIBarButtonItem) { // setup detail segue in storyboard
        performSegue(withIdentifier: "Show Detail Identifier", sender: barButton)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) { // prepare detail vc when using segues
        guard let identifier = segue.identifier else { return }

        switch identifier {
        case "Show Detail Identifier":
            let DestinationViewController = segue.Destination as! DetailViewController
            DestinationViewController.title = (sender as! UIBarButtonItem).title

        default: break
        }
    }

    private func showByNavigationController(_ barButton: UIBarButtonItem) { // show and prepare detail vc when using navigationController
        guard let DestinationViewController = storyboard?.instantiateViewController(withIdentifier: "DetailViewControllerStoryboardID") as? DetailViewController else
{ return }

        DestinationViewController.title = barButton.title
        navigationController?.pushViewController(DestinationViewController, animated: true)
    }
}
```

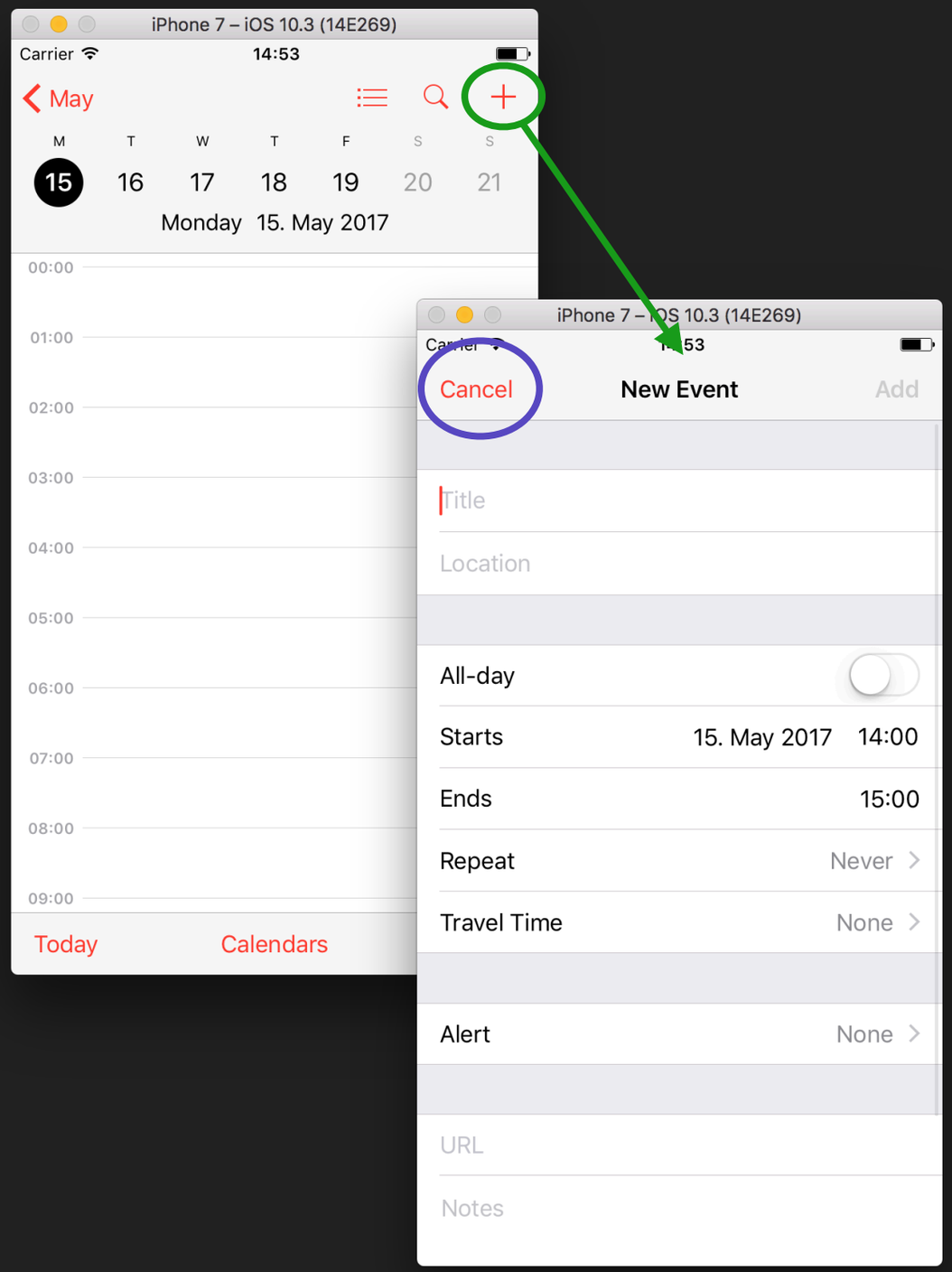
Segues

```swift
class DetailSegueViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.leftBarButtonItem = UIBarButtonItem(title: "Show Detail", style: .plain, target: self, action: #selector(showDetail(_:)))
    }

    @objc func showDetail(_ barButton: UIBarButtonItem) {
        showBySegue(barButton) // show detail vc by using segues
        showByNavigationController(barButton) // show detail vc by pushing onto navigationController stack
    }

    private func showBySegue(_ barButton: UIBarButtonItem) { // setup detail segue in storyboard
        performSegue(withIdentifier: "Show Detail Identifier", sender: barButton)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) { // prepare detail vc when using segues
        guard let identifier = segue.identifier else { return }

        switch identifier {
        case "Show Detail Identifier":
            let DestinationViewController = segue.Destination as! DetailViewController
            DestinationViewController.title = (sender as! UIBarButtonItem).title

        default: break
        }
    }

    private func showByNavigationController(_ barButton: UIBarButtonItem) { // show and prepare detail vc when using navigationController
        guard let DestinationViewController = storyboard?.instantiateViewController(withIdentifier: "DetailViewControllerStoryboardID") as? DetailViewController else
{ return }

        DestinationViewController.title = barButton.title
        navigationController?.pushViewController(DestinationViewController, animated: true)
    }
}
```

Segues

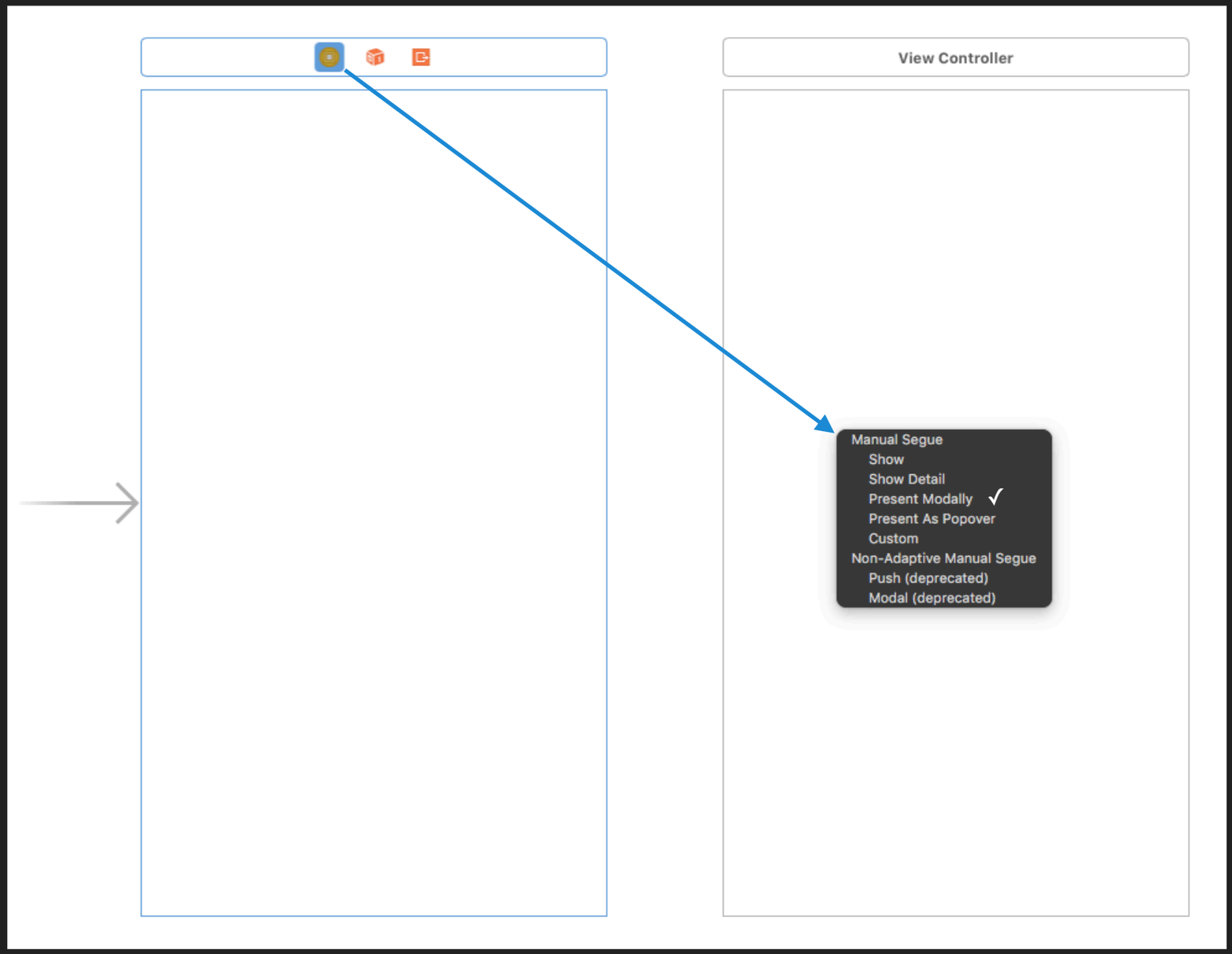
```
class DetailSegueViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.leftBarButtonItem = UIBarButtonItem(title: "Show Detail", style: .plain, target: self, action: #selector(showDetail(_:)))
    }

    @objc func showDetail(_ barButton: UIBarButtonItem) {
        showBySegue(barButton) // show detail vc by using segues
        showByNavigationController(barButton) // show detail vc by pushing onto navigationController stack
    }

    private func showBySegue(_ barButton: UIBarButtonItem) { // setup detail segue in storyboard
        performSegue(withIdentifier: "Show Detail Identifier", sender: barButton)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) { // prepare detail vc when using segues
        guard let identifier = segue.identifier else { return }

        switch identifier {
        case "Show Detail Identifier":
            let DestinationViewController = segue.Destination as! DetailViewController
            DestinationViewController.title = (sender as! UIBarButtonItem).title

        default: break
        }
    }

    private func showByNavigationController(_ barButton: UIBarButtonItem) { // show and prepare detail vc when using navigationController
        guard let DestinationViewController = storyboard?.instantiateViewController(withIdentifier: "DetailViewControllerStoryboardID") as? DetailViewController else
{ return }

        DestinationViewController.title = barButton.title
        navigationController?.pushViewController(DestinationViewController, animated: true)
    }
}
```

Segues



- Detail: Erstellt und präsentiert einen MVC, indem der VC auf den aktuellen Navigations-Stack gepushed wird
- Modal: Erstellt und präsentiert ein MVC über den gesamten Screen
- Popover: Fast das gleiche wie Modal, nur dass es beim iPad “über” den Screen gelegt wird
- Unwind: Erstellt kein neues MVC, sondern kehrt zu einem bestehenden MVC zurück
- Embed: Einen gesamten MVC-Stack innerhalb eines VC “einbetten”

Segues


View Controller
Manual Segue
Show
Show Detail
Present Modally
Present As Popover
Custom
Non-Adaptive Manual Segue
Push (deprecated)
Modal (deprecated)

Segues

```
class ModalViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.leftBarButtonItem = UIBarButtonItem(title: "Show Modal", style: .plain, target: self, action: #selector(showModal(_:)))
    }

    @objc func showModal(_ barButton: UIBarButtonItem) {
        showBySegue(barButton) // just change from detail to modal...
        showByPresenting(barButton)
    }

    private func showBySegue(_ barButton: UIBarButtonItem) { // nothing new here
        performSegue(withIdentifier: "Show Modal Identifier", sender: barButton)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) { // nothing new here
        guard let identifier = segue.identifier else { return }

        switch identifier {
        case "Show Modal Identifier":
            let DestinationViewController = segue.Destination as! DetailViewController
            DestinationViewController.title = (sender as! UIBarButtonItem).title

        default: break
        }
    }

    private func showByPresenting(_ barButton: UIBarButtonItem) {
        guard let DestinationViewController = storyboard?.instantiateViewController(withIdentifier: "DetailViewControllerStoryboardID") as? DetailViewController else
{ return }

        DestinationViewController.title = barButton.title
        present(DestinationViewController, animated: true, completion: nil)
    }
}
```

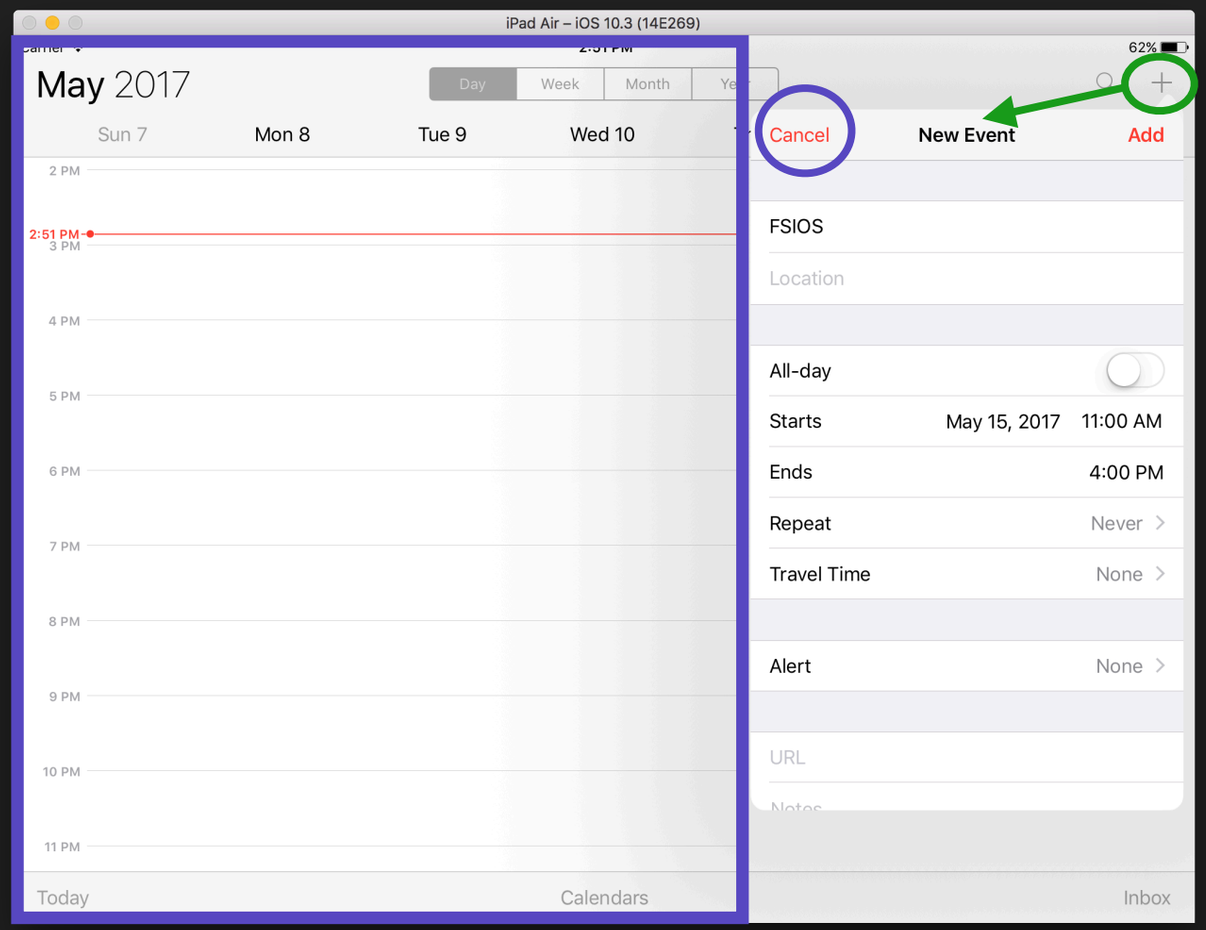
Segues

```
class ModalViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.leftBarButtonItem = UIBarButtonItem(title: "Show Modal", style: .plain, target: self, action: #selector(showModal(_:)))
    }

    @objc func showModal(_ barButton: UIBarButtonItem) {
        showBySegue(barButton) // just change from detail to modal...
        showByPresenting(barButton)
    }

    private func showBySegue(_ barButton: UIBarButtonItem) { // nothing new here
        performSegue(withIdentifier: "Show Modal Identifier", sender: barButton)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) { // nothing new here
        guard let identifier = segue.identifier else { return }

        switch identifier {
        case "Show Modal Identifier":
            let DestinationViewController = segue.Destination as! DetailViewController
            DestinationViewController.title = (sender as! UIBarButtonItem).title

        default: break
        }
    }

    private func showByPresenting(_ barButton: UIBarButtonItem) {
        guard let DestinationViewController = storyboard?.instantiateViewController(withIdentifier: "DetailViewControllerStoryboardID") as? DetailViewController else
{ return }

        DestinationViewController.title = barButton.title
        present(DestinationViewController, animated: true, completion: nil)
    }
}
```

Segues

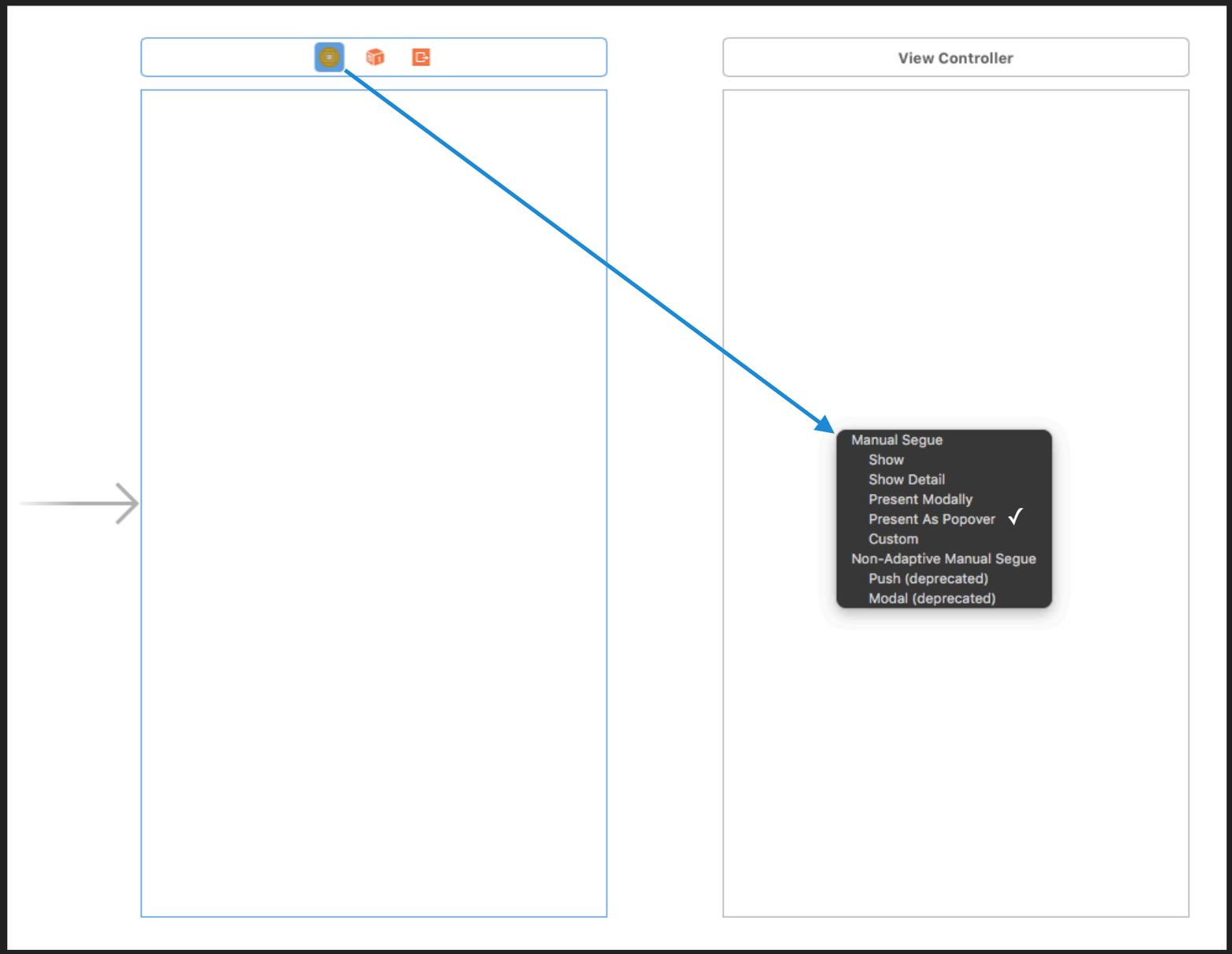
```
class ModalViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.leftBarButtonItem = UIBarButtonItem(title: "Show Modal", style: .plain, target: self, action: #selector(showModal(_:)))
    }

    @objc func showModal(_ barButton: UIBarButtonItem) {
        showBySegue(barButton) // just change from detail to modal...
        showByPresenting(barButton)
    }

    private func showBySegue(_ barButton: UIBarButtonItem) { // nothing new here
        performSegue(withIdentifier: "Show Modal Identifier", sender: barButton)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) { // nothing new here
        guard let identifier = segue.identifier else { return }

        switch identifier {
        case "Show Modal Identifier":
            let DestinationViewController = segue.Destination as! DetailViewController
            DestinationViewController.title = (sender as! UIBarButtonItem).title

        default: break
        }
    }

    private func showByPresenting(_ barButton: UIBarButtonItem) {
        guard let DestinationViewController = storyboard?.instantiateViewController(withIdentifier: "DetailViewControllerStoryboardID") as? DetailViewController else
{ return }

        DestinationViewController.title = barButton.title
        present(DestinationViewController, animated: true, completion: nil)
    }
}
```

Segues



- Detail: Erstellt und präsentiert einen MVC, indem der VC auf den aktuellen Navigations-Stack gepushed wird
- Modal: Erstellt und präsentiert ein MVC über den gesamten Screen
- Popover: Fast das gleiche wie Modal, nur dass es beim iPad “über” den Screen gelegt wird
- Unwind: Erstellt kein neues MVC, sondern kehrt zu einem bestehenden MVC zurück
- Embed: Einen gesamten MVC-Stack innerhalb eines VC “einbetten”

Segues


View Controller
Manual Segue
Show
Show Detail
Present Modally
Present As Popover ✓
Custom
Non-Adaptive Manual Segue
Push (deprecated)
Modal (deprecated)

Segues

```
class PopoverViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.leftBarButtonItem = UIBarButtonItem(title: "Show Popover", style: .plain, target: self, action: #selector(showModal(_:)))
    }

    @objc func showPopover(_ barButton: UIBarButtonItem) {
        showBySegue(barButton) // just change from modal to popover...
        showByPresenting(barButton)
    }

    private func showBySegue(_ barButton: UIBarButtonItem) { // nothing new here
        performSegue(withIdentifier: "Show Popover Identifier", sender: barButton)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) { // nothing new here
        guard let identifier = segue.identifier else { return }

        switch identifier {
        case "Show Popover Identifier":
            let DestinationViewController = segue.Destination as! DetailViewController
            DestinationViewController.title = (sender as! UIBarButtonItem).title

        default: break
        }
    }

    private func showByPresenting(_ barButton: UIBarButtonItem) {
        guard let DestinationViewController = storyboard?.instantiateViewController(withIdentifier: "DetailViewControllerStoryboardID") as? DetailViewController else { return }

        DestinationViewController.title = barButton.title

        DestinationViewController.modalPresentationStyle = .popover // change to popover
        DestinationViewController.popoverPresentationController?.barButtonItem = barButton // change source where arrow points to

        present(DestinationViewController, animated: true, completion: nil)
    }
}
```

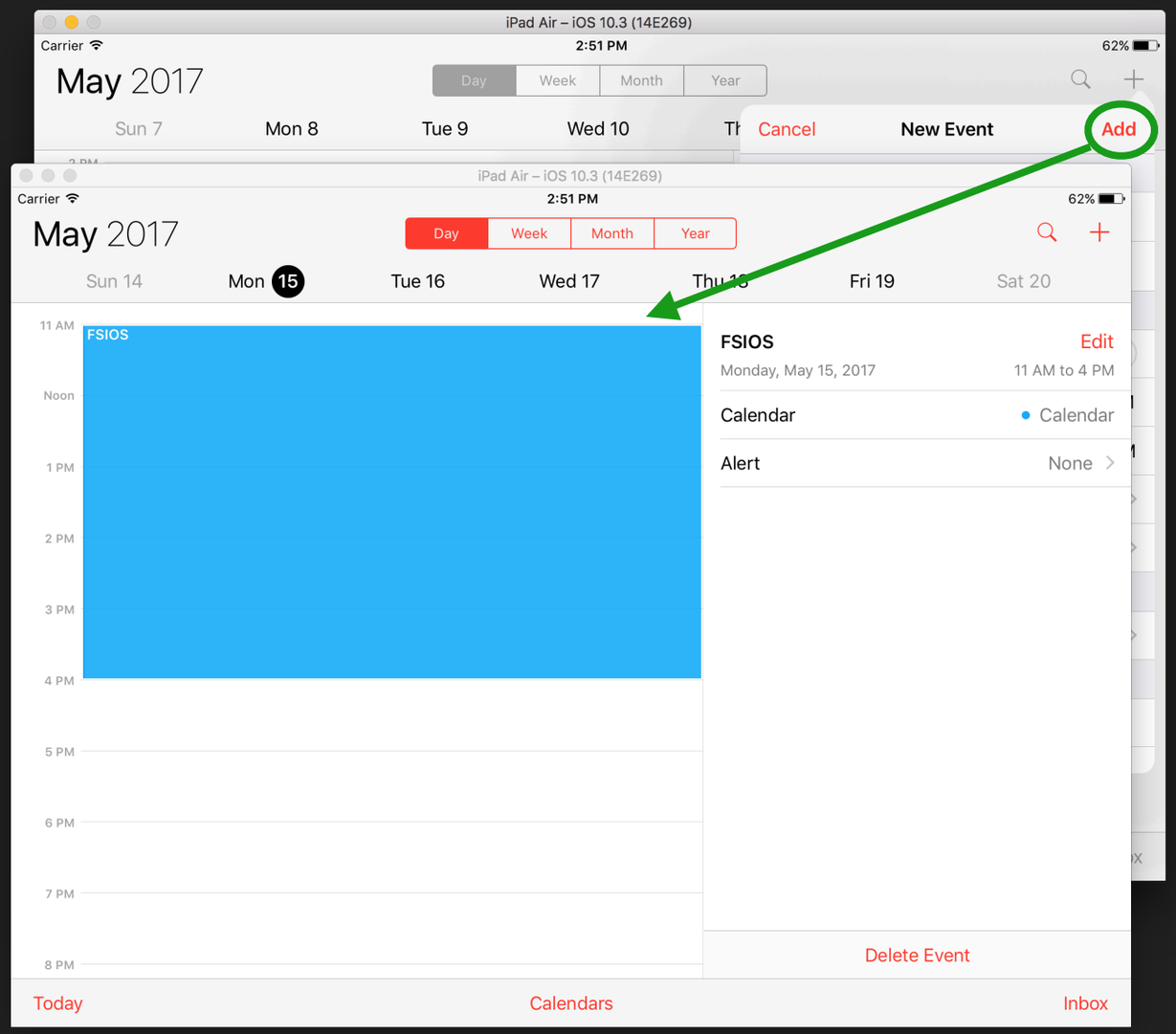
Segues

```
class PopoverViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.leftBarButtonItem = UIBarButtonItem(title: "Show Popover", style: .plain, target: self, action: #selector(showModal(_:)))
    }

    @objc func showPopover(_ barButton: UIBarButtonItem) {
        showBySegue(barButton) // just change from modal to popover...
        showByPresenting(barButton)
    }

    private func showBySegue(_ barButton: UIBarButtonItem) { // nothing new here
        performSegue(withIdentifier: "Show Popover Identifier", sender: barButton)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) { // nothing new here
        guard let identifier = segue.identifier else { return }

        switch identifier {
        case "Show Popover Identifier":
            let DestinationViewController = segue.Destination as! DetailViewController
            DestinationViewController.title = (sender as! UIBarButtonItem).title

        default: break
        }
    }

    private func showByPresenting(_ barButton: UIBarButtonItem) {
        guard let DestinationViewController = storyboard?.instantiateViewController(withIdentifier: "DetailViewControllerStoryboardID") as? DetailViewController else { return }

        DestinationViewController.title = barButton.title

        DestinationViewController.modalPresentationStyle = .popover // change to popover
        DestinationViewController.popoverPresentationController?.barButtonItem = barButton // change source where arrow points to

        present(DestinationViewController, animated: true, completion: nil)
    }
}
```

Segues

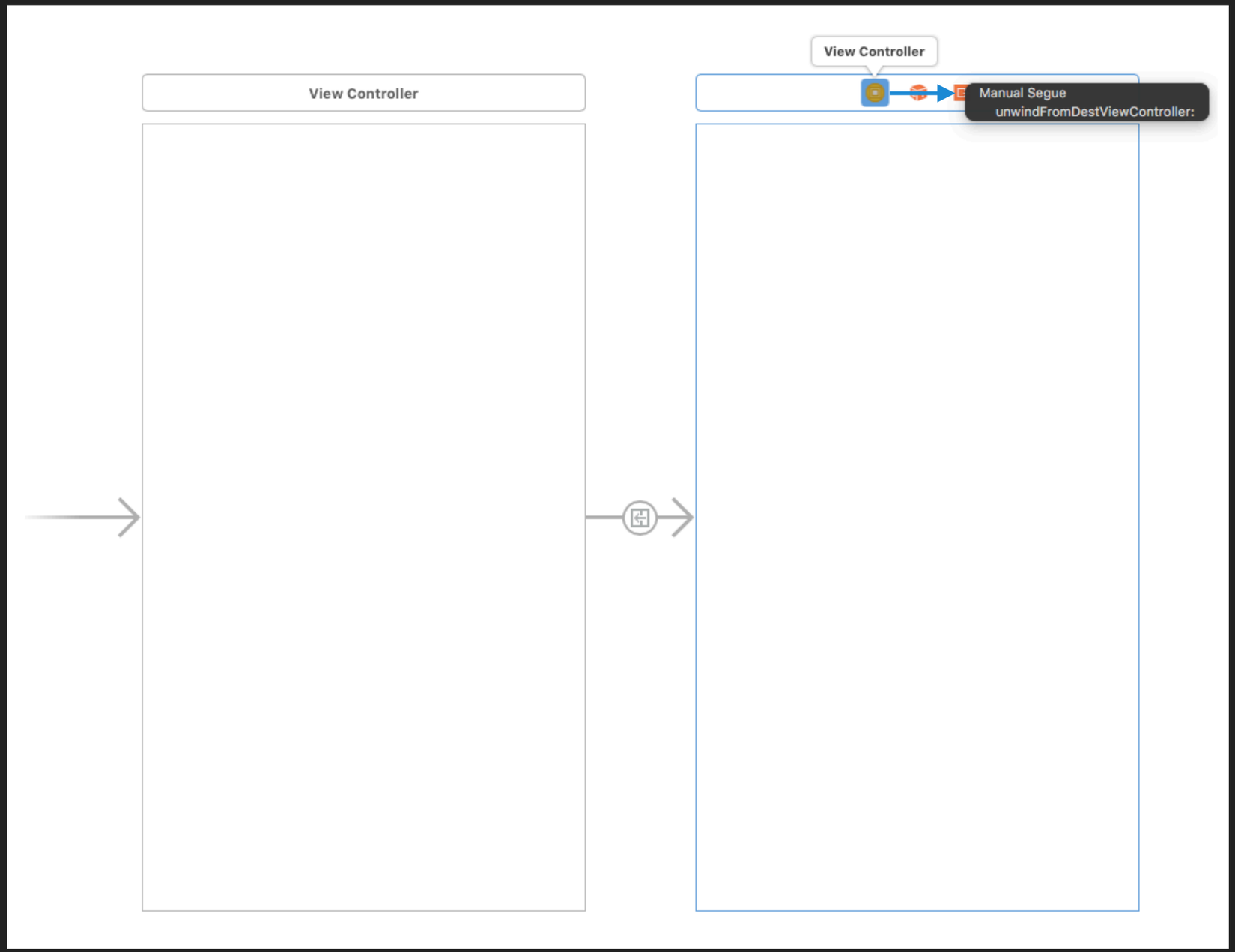
```
class PopoverViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.leftBarButtonItem = UIBarButtonItem(title: "Show Popover", style: .plain, target: self, action: #selector(showModal(_:)))
    }

    @objc func showPopover(_ barButton: UIBarButtonItem) {
        showBySegue(barButton) // just change from modal to popover...
        showByPresenting(barButton)
    }

    private func showBySegue(_ barButton: UIBarButtonItem) { // nothing new here
        performSegue(withIdentifier: "Show Popover Identifier", sender: barButton)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) { // nothing new here
        guard let identifier = segue.identifier else { return }

        switch identifier {
        case "Show Popover Identifier":
            let DestinationViewController = segue.Destination as! DetailViewController
            DestinationViewController.title = (sender as! UIBarButtonItem).title

        default: break
        }
    }

    private func showByPresenting(_ barButton: UIBarButtonItem) {
        guard let DestinationViewController = storyboard?.instantiateViewController(withIdentifier: "DetailViewControllerStoryboardID") as? DetailViewController else { return }

        DestinationViewController.title = barButton.title

        DestinationViewController.modalPresentationStyle = .popover // change to popover
        DestinationViewController.popoverPresentationController?.barButtonItem = barButton // change source where arrow points to

        present(DestinationViewController, animated: true, completion: nil)
    }
}
```

Segues



- Detail: Erstellt und präsentiert einen MVC, indem der VC auf den aktuellen Navigations-Stack gepushed wird
- Modal: Erstellt und präsentiert ein MVC über den gesamten Screen
- Popover: Fast das gleiche wie Modal, nur dass es beim iPad “über” den Screen gelegt wird
- Unwind: Erstellt kein neues MVC, sondern kehrt zu einem bestehenden MVC zurück
- Embed: Einen gesamten MVC-Stack innerhalb eines VC “einbetten”

Segues


View Controller
View Controller
Manual Segue
unwindFromDestViewController:

Segues

```swift
class SrcViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        performSegue(withIdentifier: "Some Identifier", sender: self)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        guard segue.identifier == "Some Identifier", let DestViewController = segue.Destination as? DestViewController else { return }
        DestViewController.model = "new model from SrcViewController"
    }

    @IBAction func unwindFromDestViewController(_ segue: UIStoryboardSegue) { // this is called on SrcViewController when unwinding happens
        guard let SrcViewController = segue.source as? DestViewController else { return }
        print(SrcViewController.model)
    }
}

class DestViewController: UIViewController {

    var model: String? // set by SrcViewController first. updated by DestViewController later on

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.rightBarButtonItem = UIBarButtonItem(barButtonSystemItem: .done, target: self, action: #selector(done))
    }

    @objc func done() {
        performSegue(withIdentifier: "Unwind Segue", sender: self) // setup unwind segue in storyboard first ...
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        guard segue.identifier == "Unwind Segue" else { return }
        model = "updated model from DestViewController"
    }
}
```

Segues

```swift
class SrcViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        performSegue(withIdentifier: "Some Identifier", sender: self)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        guard segue.identifier == "Some Identifier", let DestViewController = segue.Destination as? DestViewController else { return }
        DestViewController.model = "new model from SrcViewController"
    }

    @IBAction func unwindFromDestViewController(_ segue: UIStoryboardSegue) { // this is called on SrcViewController when unwinding happens
        guard let SrcViewController = segue.source as? DestViewController else { return }
        print(SrcViewController.model)
    }
}

class DestViewController: UIViewController {

    var model: String? // set by SrcViewController first. updated by DestViewController later on

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.rightBarButtonItem = UIBarButtonItem(barButtonSystemItem: .done, target: self, action: #selector(done))
    }

    @objc func done() {
        performSegue(withIdentifier: "Unwind Segue", sender: self) // setup unwind segue in storyboard first ...
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        guard segue.identifier == "Unwind Segue" else { return }
        model = "updated model from DestViewController"
    }
}
```

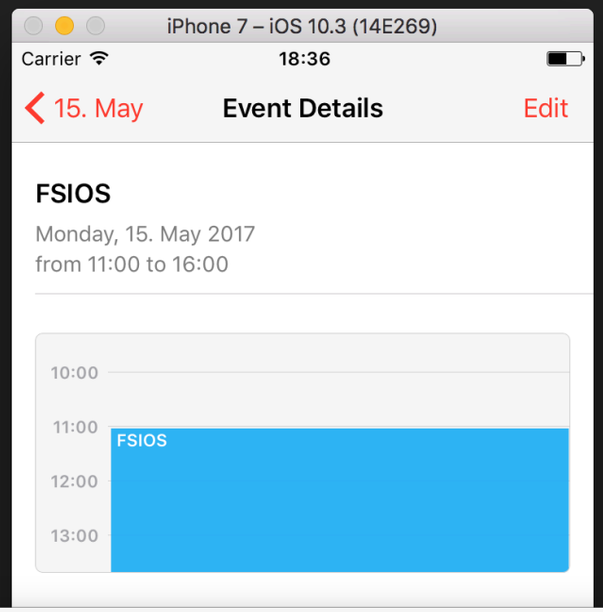
Segues

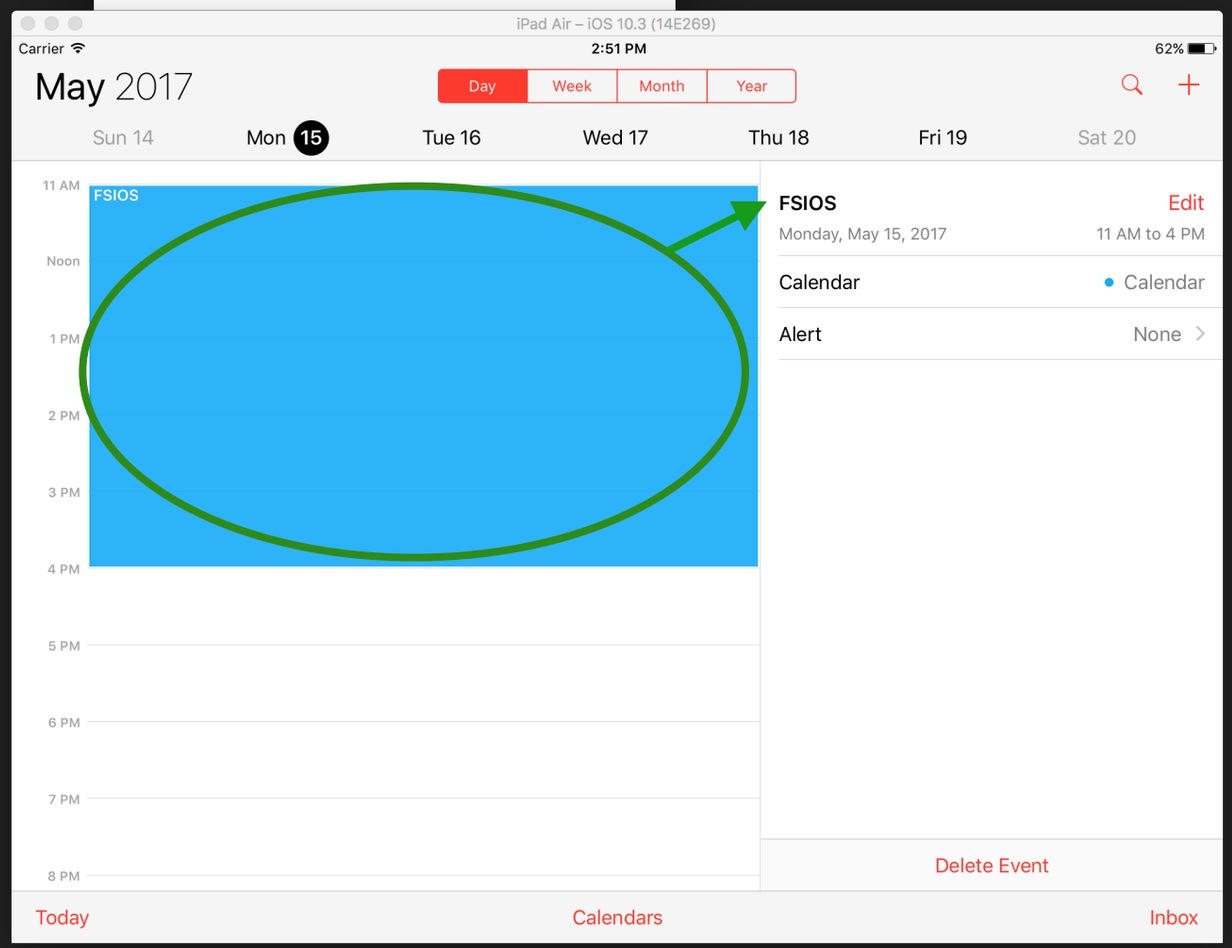
```
class SrcViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        performSegue(withIdentifier: "Some Identifier", sender: self)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        guard segue.identifier == "Some Identifier", let DestViewController = segue.Destination as? DestViewController else { return }
        DestViewController.model = "new model from SrcViewController"
    }

    @IBAction func unwindFromDestViewController(_ segue: UIStoryboardSegue) { // this is called on SrcViewController when unwinding happens
        guard let SrcViewController = segue.source as? DestViewController else { return }
        print(SrcViewController.model)
    }
}

class DestViewController: UIViewController {

    var model: String? // set by SrcViewController first. updated by DestViewController later on

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.rightBarButtonItem = UIBarButtonItem(barButtonSystemItem: .done, target: self, action: #selector(done))
    }

    @objc func done() {
        performSegue(withIdentifier: "Unwind Segue", sender: self) // setup unwind segue in storyboard first ...
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        guard segue.identifier == "Unwind Segue" else { return }
        model = "updated model from DestViewController"
    }
}
```

Segues

```swift
class SrcViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        performSegue(withIdentifier: "Some Identifier", sender: self)
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        guard segue.identifier == "Some Identifier", let DestViewController = segue.Destination as? DestViewController else { return }
        DestViewController.model = "new model from SrcViewController"
    }

    @IBAction func unwindFromDestViewController(_ segue: UIStoryboardSegue) { // this is called on SrcViewController when unwinding happens
        guard let SrcViewController = segue.source as? DestViewController else { return }
        print(SrcViewController.model)
    }
}

class DestViewController: UIViewController {

    var model: String? // set by SrcViewController first. updated by DestViewController later on

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.rightBarButtonItem = UIBarButtonItem(barButtonSystemItem: .done, target: self, action: #selector(done))
    }

    @objc func done() {
        performSegue(withIdentifier: "Unwind Segue", sender: self) // setup unwind segue in storyboard first ...
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        guard segue.identifier == "Unwind Segue" else { return }
        model = "updated model from DestViewController"
    }
}
```

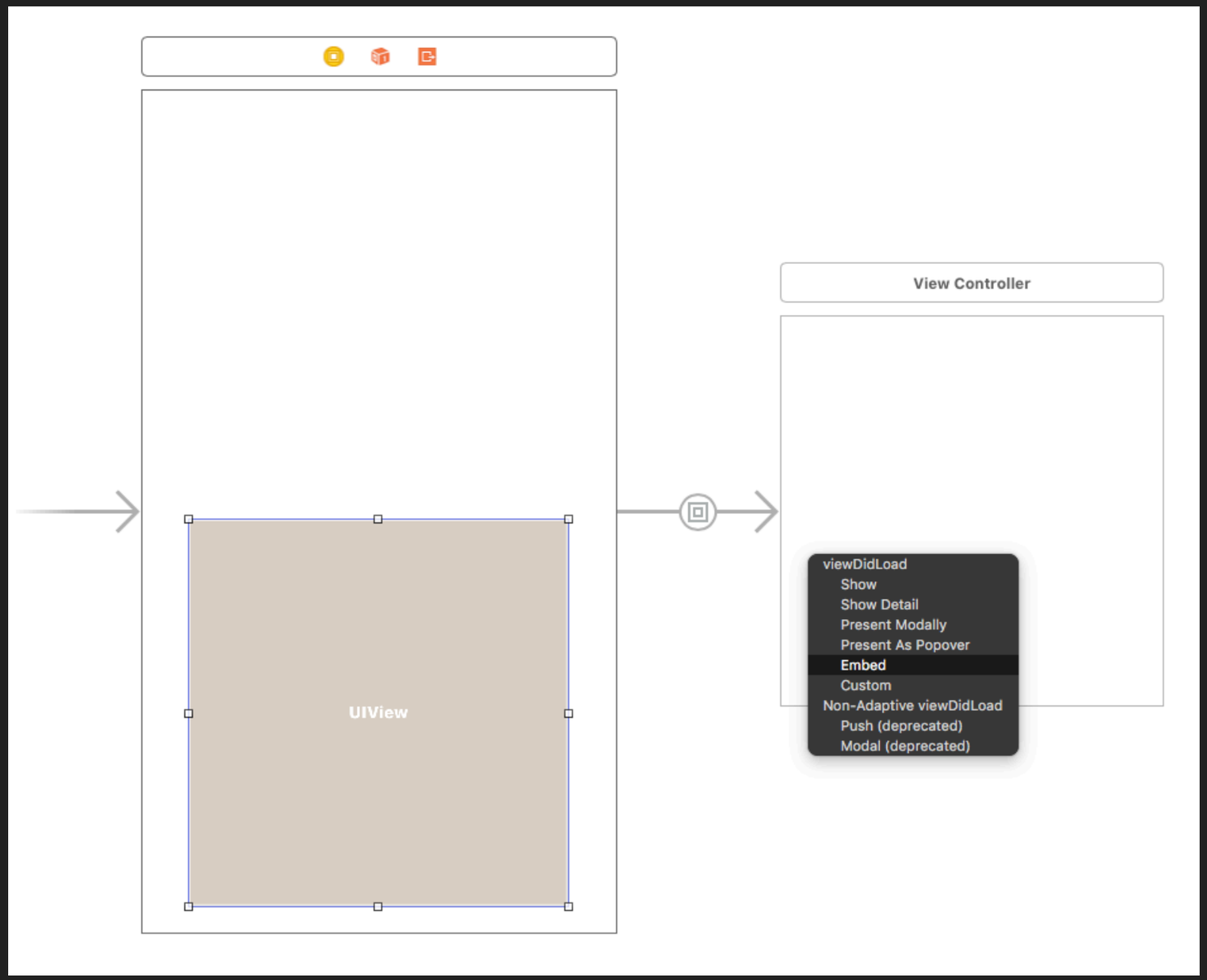
Segues





- Detail: Erstellt und präsentiert einen MVC, indem der VC auf den aktuellen Navigations-Stack gepushed wird
- Modal: Erstellt und präsentiert ein MVC über den gesamten Screen
- Popover: Fast das gleiche wie Modal, nur dass es beim iPad “über” den Screen gelegt wird
- Unwind: Erstellt kein neues MVC, sondern kehrt zu einem bestehenden MVC zurück
- Embed: Bettet einen gesamten MVC-Stack in einen anderen VC ein

Segues


View Controller
UIView
viewDidLoad
Show
Show Detail
Present Modally
Present As Popover
Embed
Custom
Non-Adaptive viewDidLoad
Push (deprecated)
Modal (deprecated)

Segues

```
class ViewControllerToReuse: UIViewController { }

class EmbedSegueViewController: UIViewController {

    var viewControllerToReuse: ViewControllerToReuse?

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        guard segue.identifier = "Embed Segue" else { return }

        viewControllerToReuse = segue.Destination as? ViewControllerToReuse
    }
}
```

Segues

- Zur Rückkommunikation von DestVC zum SrcVC gibt es unterschiedliche Möglichkeiten
    - Uwind-Segue, welche eine @IBAction Funktion des SrcVC aufruft und dort die Möglichkeit gibt, auf den DestVC zuzugreifen, ähnlich zu prepare(for segue:, identifier:)
    - DestVC hat ein eigenes Delegate und ruft diesen auf in viewWillDisappear oder navigationController(_ willShow:, animated:) // “<“ in der NavigationBar
    - Notification Pattern, wo der DestVC das Ergebnis in die Datenbank speichert und der SrcVC durch Observieren benachrichtigt wird
- Die Auswahl der Möglichkeit sollte sich nach dem Kontext des DestVC richten*
    - DestVC wurde in einer Modal-Segue präsentiert: Modal-Segues haben keine hierarchische Struktur zum SrcVC. Zudem werden sie FullScreen präsentiert, weshalb es keinen Zurück-Button gibt -> z.B. Unwind-Segue
    - DestVC wurde in einer Detail-Segue präsentiert: Hierarchische Struktur kann zum Verlassen des Dest-VC genutzt werden -> z.B. Delegate
    - Sollen Daten zurückgesendet und der DestVC frühzeitig verlassen werden? -> Unwind-Segue
    - Gibt es mehr als einen Grund zum Zurücksenden? -> Delegate oder ein Enum UnwindReason mit Unwind-Segue
- Die Unwind-Segue löst ein implizites Verlassen des DestVC aus. Sonstige Möglichkeiten sind
    - presentingViewController?.dismiss(animated: true, completion: nil) für Modal- und
    - navigationController?.popViewController(animated: true) für Detail-Segues

*Das sind lediglich Empfehlungen. Theoretisch kann man alles verwenden. Einiges ist geeigneter als anders.

Heute

Application Lifecycle
ViewController Lifecycle
Segues

Zusammenfassung

Zusammenfassung

- iOS verfügt über einen Application- und ViewController-Lifecycle, um dem Entwickler die Möglichkeit zu geben, in bestimmten Situationen angemessen zu reagieren
    - Die Kunst ist es, für ein entsprechendes Problem die richtigen Lifecycle Funktionen zu verwenden
    - Die Klasse AppDelegate soll ausschließlich als Delegate verwendet werden und keinen State beinhalten
    - Viele Funktionen von AppDelegate lassen sich mit NotificationCenter observieren
- Segues ermöglichen Transitionen zwischen ViewController
    - Segues sind der empfohlene Weg, allerdings können die ViewController auch im Code instanziiert und präsentiert, eingebettet oder gepushed werden
    - Es gibt unterschiedliche Möglichkeiten, um einen präsentierten bzw. gepushten ViewController zu verlassen
    - Modal präsentierte ViewController beanspruchen die vollständige Aufmerksamkeit des Benutzers
    - Je abgegrenzter und strukturierter ein ViewController ist, Desto einfacher lässt er sich in andere ViewController einbetten und wiederverwenden
    - Popover-Segues werden von iPhone und iPad automatisch adaptiert, wodurch die User-Experience beim iPad angenehmer ist
- Die Vorbereitung eines ViewController und das Setzen der Outlets ist ein Teil des ViewController-Lifecycle und muss unbedingt beachtet werden