iOS Foundations II
Session 2

- Git
- Initializers
- Optionals?
- View Controllers
- Segues
- Navigation Controllers

What is Git?

- Git is an open source version control system.
- Git is an application you install on your computer.
- Git itself is not Github, Github is a hosting service for git repositories.
- type git —version into terminal to see if you have git installed.

Git Basics

- git init - Creates an empty git repo or reinitializes an existing one.
- After running git init, there will be a hidden .git file in the directory you are in. type ls -a to list all files. You will see the .git file.
- this .git file is a directory that takes snapshots of your project's files every time you commit or save the state of your project.

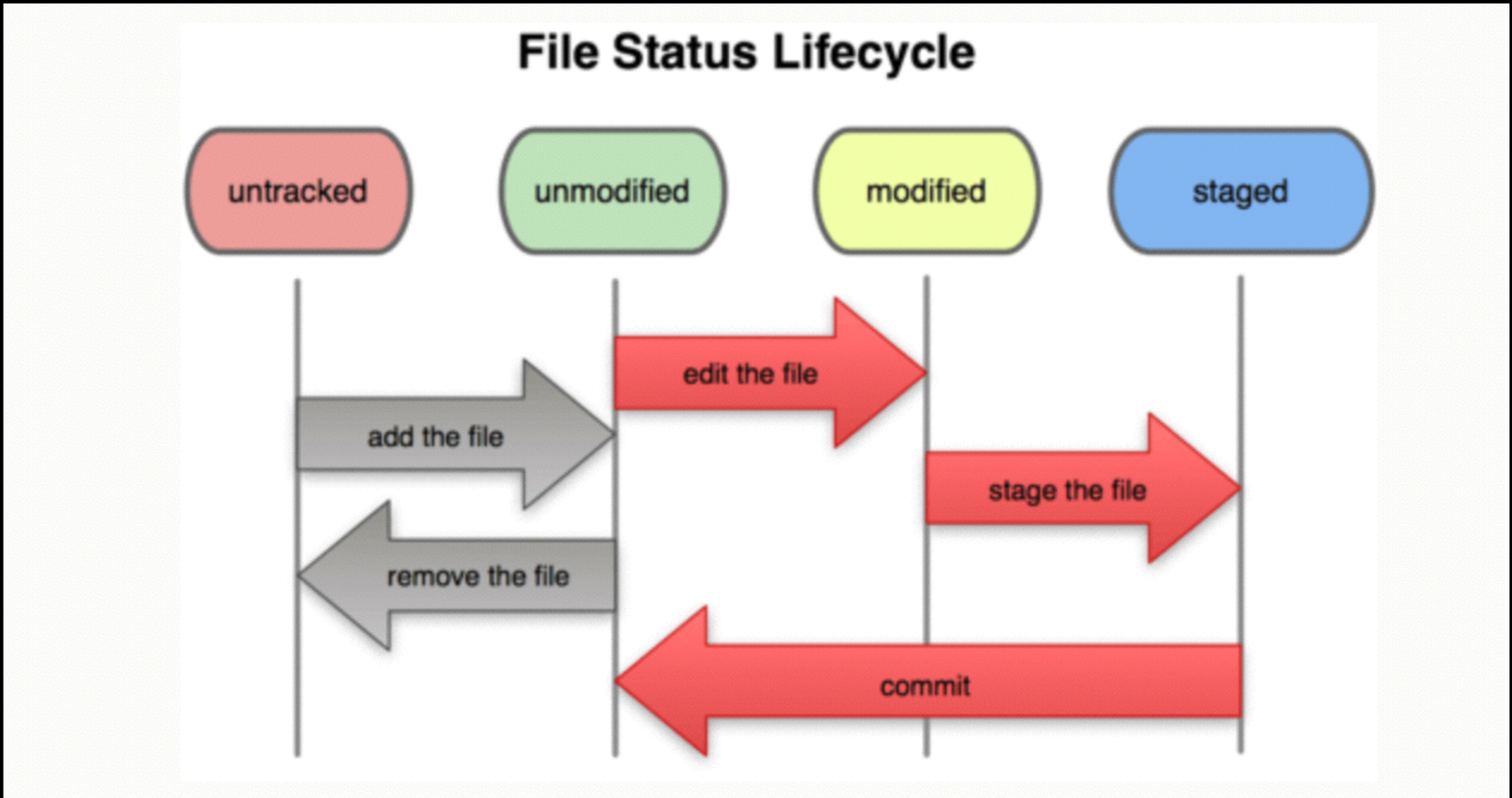
Git Basics

- the .git files keeps track of everything by tracking 3 sections:
- Working Directory - a single checkout of one version of the project. These files are uncompressed after being pulled out of the object database and placed on disk for use.
- Staging area - one file, stores information about what will go into your next commit.
- Git Directory - where git stores the metadata and object database of your project

Git Workflow

1. You modify files in your working directory.
2. You stage the files, which adds snapshots of them to your staging area.
3. You do a commit, which takes files as they are in the staging area and stores snapshot permanently to your git directory.
4. Profit

Git File Status Lifecycle



Git Basics

- Use the git status command to determine the state of your files
- It also tells you which branch you are on
- use git add to begin tracking new files and to stage files after they have been modified
- use git add to also mark a merge conflict as resolved, more about this later.
- Once you stage all your changes with git add, use git commit to commit those changes.
- git commit -m “commit message” is quicker and cleaner than git commit

Git Remotes

- Remote repos are versions of your project that are hosted in the Cloud™ or a private network.
- Run the git remote command to see if you have any remote servers configured.
- Cloning a repo automatically adds that remote repo under the name origin. (git clone <url>)
- Origin is the default name of a remote repo, but you can call it whatever you want

Git Remote Commands

- git remote -v shows the URL for each remote as well
- use git remote add <nickname> <url> to add a remote repo
- after committing, use git push <nickname> <branch> to push your committed changes to your remote
- use git pull to automatically fetch and merge a remote branch into your current branch
- Demo!

Git and Xcode

- Once you initialize a repository and add your remote repositories to an Xcode project, you can use Xcode's Source Control as an alternative to using the command line.
- You can push,pull,commit, switch branches, basically everything you would have done in the command line.

Initializers

- properties must be initialized one way or another
- Swift does not allow you to leave properties in an undetermined state.
- variables must either:
  - be given a default value
  - have their value set in the initializer
  - be marked as optional using either the ? or ! symbol

Optionals?

- forced unwrapping (!) accesses the value inside an optional
- optional binding checks if it has a value before proceeding
- optional chaining validates the optional nested properties and aborts after first nil?
- use implicitly unwrapped optionals if you can ensure an optional var will have a value before it is first accessed
- conditionally downcast to an `if let` using `as?`
  - `if let firstName = person[“firstName”] as? String { … }`

ViewControllers

- Basic Visual Component of iOS.
- Everything onscreen is managed by ViewControllers and their subviews.They are the link between your model layer and view layer.
- iOS provides many built-in ViewControllers to support standard interfaces.
- The most important property of a ViewController is it’s view property.

ViewControllers

- Really only 4 ways to get a ViewController's view on screen:
    1. Make the ViewController a window's rootViewController.
    2. Make the ViewController a child of a container View Controller
    3. Show the ViewController in a popover
    4. Present it from another View Controller (The most common scenario)

ViewController Communication

- There are many ways ViewControllers can communicate with each other, and this particular topic is often hard to grasp at first.
- Delegation and notifications are common techniques, but an even simpler way is just passing a reference to the necessary model object that a particular child ViewController will need.
- Remember, ViewController are objects just like any other object in Swift, they can be held in Arrays or Dictionaries, and passed around in methods or held as properties.

ViewController Life Cycle

- **The concept of ViewController's lifecycle is important to understand. So important we bolded it.**
- There are 5 methods to know:
    1. ViewDidLoad - Called when the ViewController's view is loaded into memory
    2. ViewWillAppear - Called before the ViewController's view appears onscreen
    3. ViewDidAppear - Called immediately after the ViewController's view has appeared onscreen
    4. ViewWillDisappear - Called when the ViewController's view is about to be removed from the view hierarchy

Segues

- Segues are provided by the storyboard to help you easily transition from one view controller to another.
- There are 2 primary segues that are used : Show and Present
- Show refers to pushing a view controller onto the navigation stack, it usually will slide in from the right or left.
- Present refers to modally presenting a view controller, it usually slides up from the bottom.
- You can create your own custom segue to customize the behavior to match your needs.

Segues

- 2 primary methods for dealing with segues in code:
    1. performSegueWithIdentifier:
    2. prepareForSegue:sender:

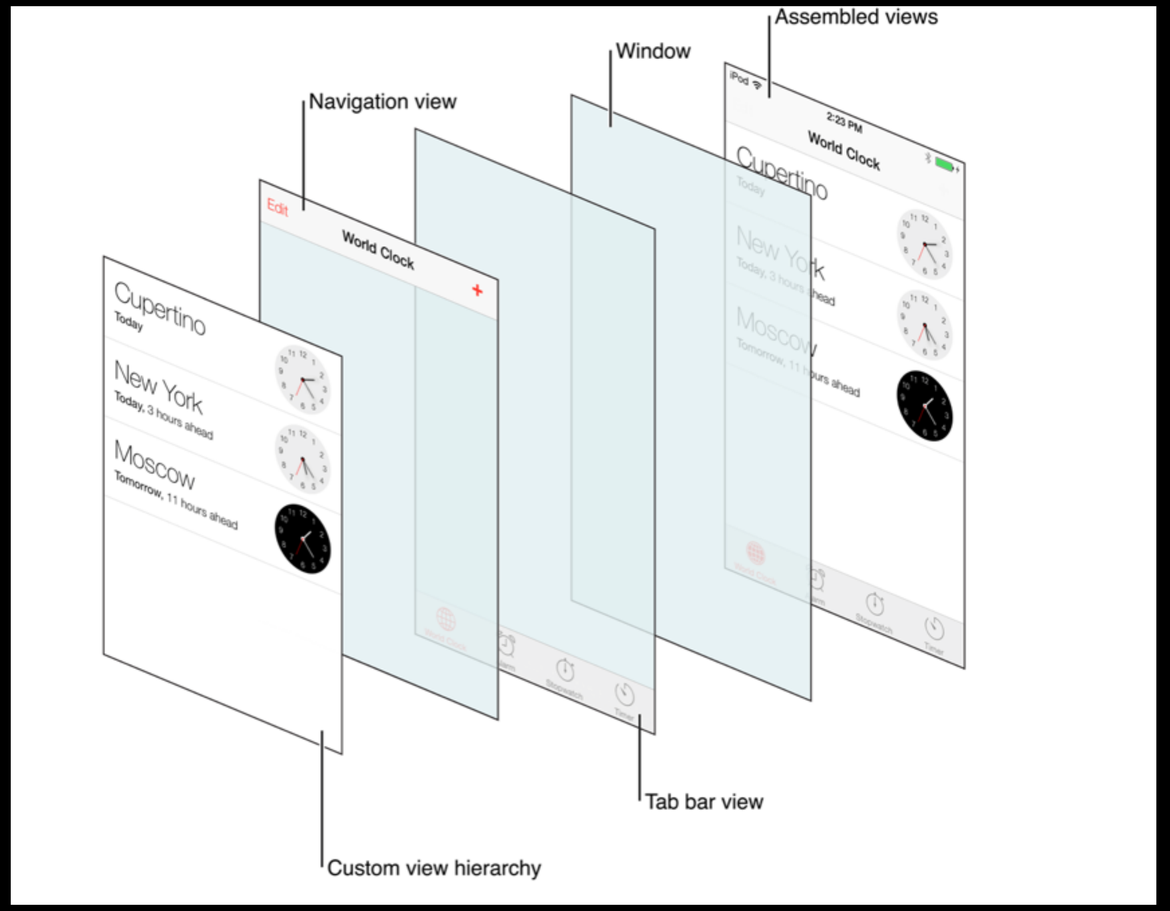
performSegueWithIdentifier

- Call this method on in your View Controller to trigger a segue in code.
- If you your segue isn't hooked up to be triggered by an action in your interface, you will need to call this method to trigger the segue.

prepareForSegue:sender:

- This method is called on the source view controller of the segue, right before the segue is actually performed.
- The first paramater is the segue itself, which is an instance of the UISegue class.
- The most important property of a UISegue instance is the destinationViewController property, which gives you a reference to the view controller you are about to segue to.
- This is a great spot to pass information to the next screen.

NavigationControllers



- “A navigation controller manages a stack of view controllers to provide a drill-down interface for hierarchical content.”

NavigationControllers

- “The navigation controller’s primary responsibility is to respond to user actions by pushing new content view controllers onto the stack or popping content view controllers off of the stack”
- The first ViewController you push onto the stack becomes the rootviewcontroller and is never popped off because then no view would be on screen
- Has a property to its topViewController and and array property for all its viewControllers.
- The nav bar up top can be customized or hidden. Is 44 points tall.

NavigationControllers

- Storyboards make navigation controllers extremely easy to install into your app, heres all the methods you need to do it in code:
- `init(rootViewController:)` UINavigationController is initialized with a rootviewController.
- `pushViewController(animated:)` To add or ‘push’ a view controller onto the stack.
- `popViewController(animated:)` To remove or ‘pop’ a view controller from the stack.

Stack Data Structure

- “A stack is particular kind of abstract data type or collection in which the only operations on the collection is adding (push) or removal (pop).” - Wikipedia
- LIFO : Last-In-First-Out. The last item added is the first to be removed.