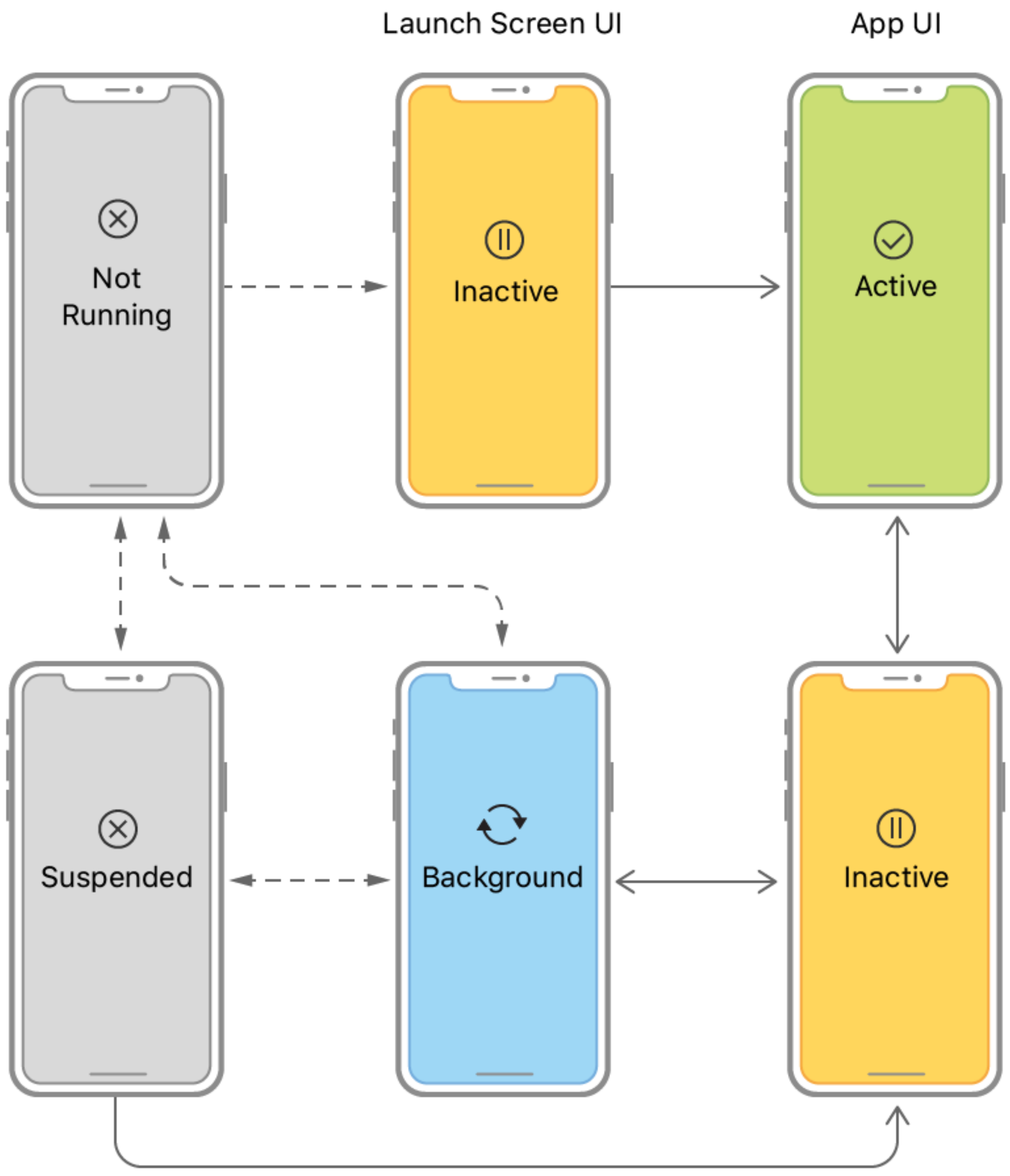
iOS Background Transfer

Simon Wang

Download & Upload

- What happens when you get a phone call or press home button?


Launch Screen UI
App UI
Not Running
Inactive
Active
Suspended
Background
Inactive

Suspended immediately!

But…

- We need data prepared in the background
- We want the data download/upload work even we are not using the app

Background task

```
beginBackgroundTask(expirationHandler:)
```

~ 180 seconds

But…

- Sometimes, 180 seconds is not enough to download/ upload all the data
    - Upload 200 photos to MEGA/Dropbox/OneDrive
    - Download 10 videos from MEGA/Dropbox/OneDrive
- What if the WiFi gets dropped and you have no network connection
- What if you don’t want it work in background because your battery is low

Background Transfer

- Transfers are managed by iOS system daemon
- Works beyond the 180 seconds
- Works even when app gets suspended or terminated
- Smart
    - Only when device is in charging
    - Only when WiFi is available
    - Wait for weeks before expire

Lifecycle - setup

1. Create background NSURLSession instance
2. Requires a delegate for event delivery, no blocks
3. Submit download/upload NSURLSessionTasks

Lifecycle - foreground

- Just a normal NSURLSession task when it finishes in foreground

Lifecycle - app not running

- When tasks finish while app is not running.

```
application(_:handleEventsForBackgroundURLSessi
on:completionHandler:)
```

- When tasks finish while app is suspended, iOS system will reconnect everything

- When tasks finish while app is not launched, you need to manually reconstitute

Lifecycle - app launches

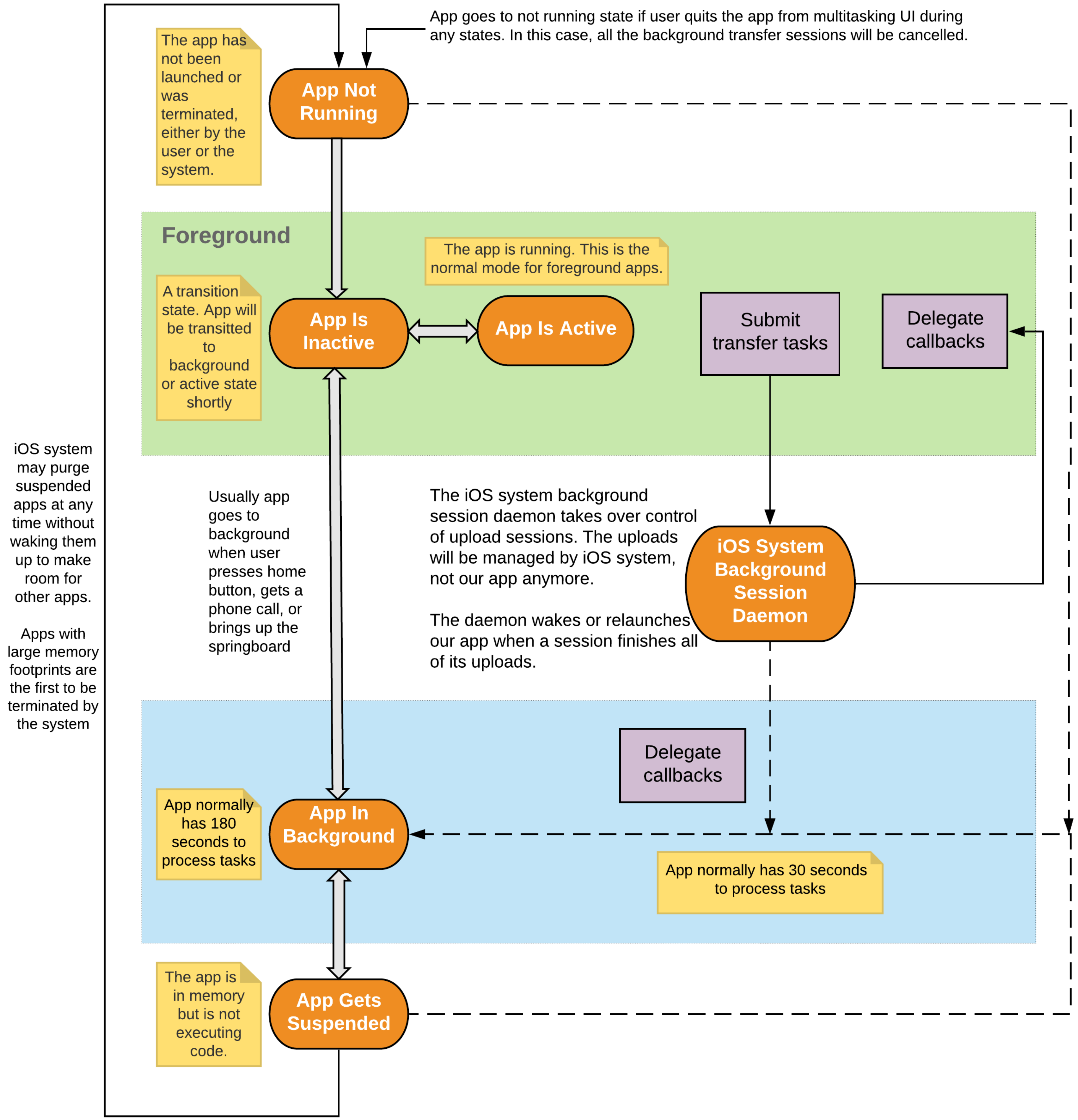
- When app launches before all the background tasks finish

Lifecycle - tasks finish

- When all the background tasks finish.

```
urlSessionDidFinishEvents(forBackgroundURLSessi
on:)
```

- Call CompletionHandler in main thread


App goes to not running state if user quits the app from multitasking UI during any states. In this case, all the background transfer sessions will be cancelled.
The app has not been launched or was terminated, either by the user or the system.
App Not Running
Foreground
The app is running. This is the normal mode for foreground apps.
A transition state. App will be transitted to background or active state shortly
App Is Inactive
App Is Active
Submit transfer tasks
Delegate callbacks
iOS system may purge suspended apps at any time without waking them up to make room for other apps.
Apps with large memory footprints are the first to be terminated by the system
Usually app goes to background when user presses home button, gets a phone call, or brings up the springboard
The iOS system background session daemon takes over control of upload sessions. The uploads will be managed by iOS system, not our app anymore.
The daemon wakes or relaunches our app when a session finishes all of its uploads.
iOS System Background Session Daemon
Delegate callbacks
App normally has 180 seconds to process tasks
App In Background
App normally has 30 seconds to process tasks
The app is in memory but is not executing code.
App Gets Suspended

Lifecycle in code

```swift
// Setup
let config = URLSessionConfiguration.background(withIdentifier:
"nz.mega.MySession")
let delegate = MyURLSessionDelegate() // setup delegate
let session = URLSession(configuration: config, delegate: delegate,
delegateQueue: nil)
let task = session.uploadTask(with: request, fromFile: url)
task.resume()

// UIApplicationDelegate
func application(_ application: UIApplication,
                 handleEventsForBackgroundURLSession identifier: String,
                 completionHandler: @escaping () -> Void) {
    //
}

// UIApplicationDelegate
func application(_ application: UIApplication,
                 willFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey : Any]? = nil) -> Bool {
    //
}

// URLSessionDelegate
func urlSessionDidFinishEvents(forBackgroundURLSession session:
URLSession) {
    //
}
```

Third party?

- AFNetworking
- Alamofire

Key Takeaways

1. Use background transfer when to download/upload data
2. Always use background tasks to extend background execution time.
3. This is terribly important when app gets launched in background. Otherwise you can not get the 30 seconds.
4. Make your code ready to handle unexpected termination
5. Good for big files
6. Don’t submit too many transfer tasks (<600)

How to cancel background transfer tasks?

- Kill the app from multitasking UI. This also prevents app launch to background.
- Cancel tasks from code
- When the resource expires

Can iOS recover the background transfer tasks upon device restarts?

- Yes, iOS background daemon can persist the background tasks to disk and recover them after device reboot.

Thanks!

&

Questions?