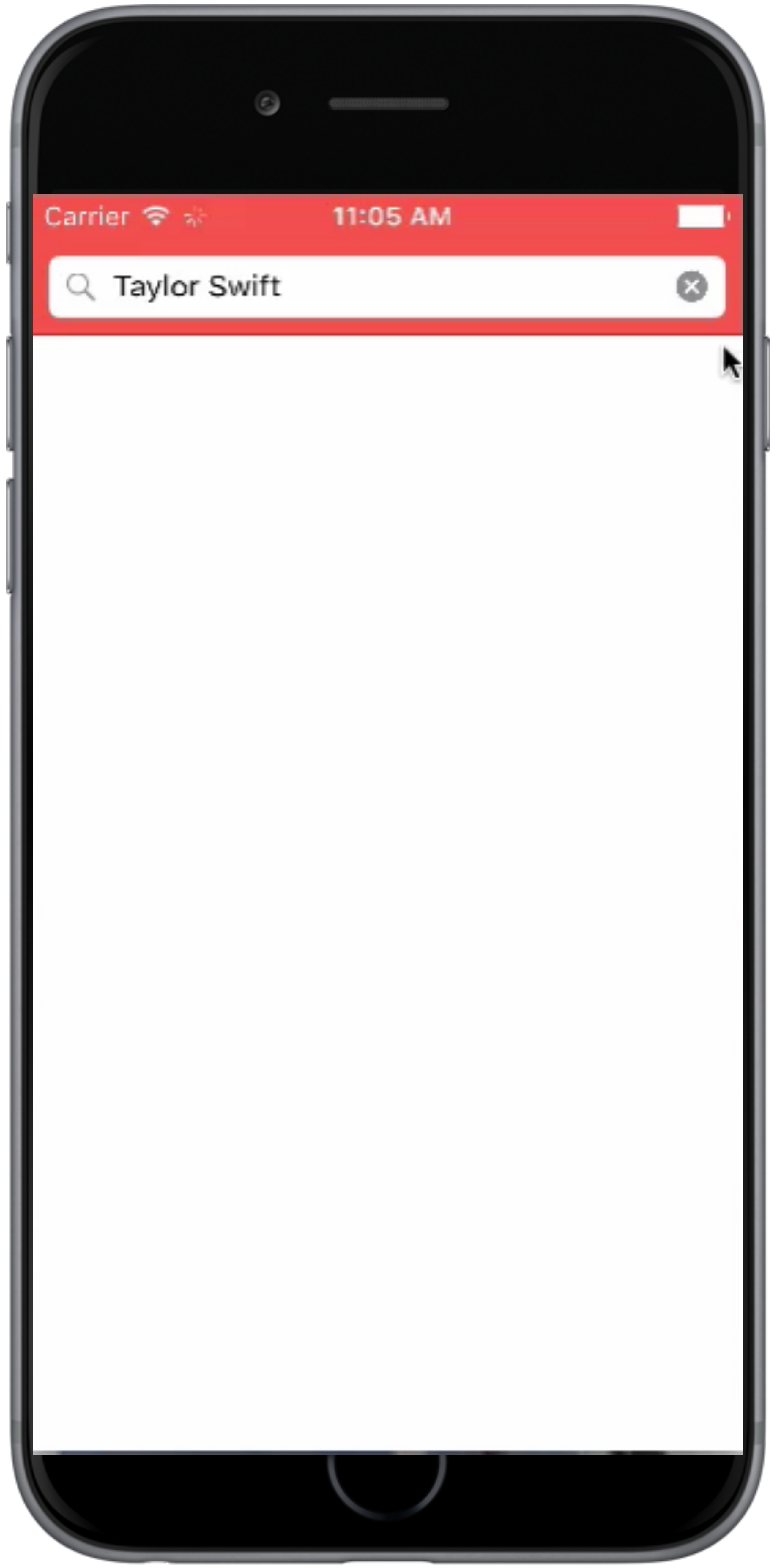
Networking with URLSession

Part 7: Background Sessions


Carrier
11:05 AM
Taylor Swift

Background Session

```
let configuration = URLSessionConfiguration.background(
  withIdentifier: “com.raywenderlich.prefetch”)
// if tasks should mainly run in the background
configuration.networkServiceType = .background
```

```
init(configuration: URLSessionConfiguration,
  delegate: URLSessionDelegate?,
  delegateQueue: OperationQueue?)
```

```
let session = URLSession(configuration: configuration,
  delegate: self, delegateQueue: nil)
```

Background Session Workflow

OS terminates app in

background events need handling

app relaunched

```
application:handleEventsForBackgroundURLSession:completionHandler:
```

store completionHandler
create background configuration with identifier
create session & handle events

```
urlSessionDidFinishEvents(forBackgroundURLSession: URLSession)
  calls stored completionHandler
```

app suspended

Background Session Considerations

- The number of system-wide concurrent background transfers is limited.
- A background task may be cancelled if it fails to meet a system-specified throughput limit.
- If the background transfer is initiated while the app is in the background, the task is treated as discretionary.
- Redirects are always followed.
- Upload tasks must be from a file.

Server-side Negotiations

Set up your back-end server with endpoints that:

- send or receive zip or tar archives
- send or receive incremental diffs for replication between the client and server
- return an upload identifier, which your app can use to track and resume the upload of data

Need Custom Delegate If App:

- Uses background sessions to download or upload content while it’s not running.
- Performs custom authentication or SSL certificate verification.
- Decides whether a transfer should be downloaded to disk or displayed based on the MIME type or other response info.
- Limits caching or HTTP redirects programmatically.

Session Delegate Methods

URLSessionDelegate

```
func urlSession(URLSession, didBecomeInvalidWithError: Error?)
func urlSessionDidFinishEvents(forBackgroundURLSession: URLSession)
```

URLSessionTaskDelegate

```
func urlSession(URLSession, task: URLSessionTask,
  didSendBodyData: Int64, totalBytesSent: Int64,
  totalBytesExpectedToSend: Int64)
func urlSession(URLSession, task: URLSessionTask,
  didCompleteWithError: Error?)
func urlSession(URLSession, task: URLSessionTask,
  willPerformHTTPRedirection: HTTPURLResponse,
  newRequest: URLRequest,
  completionHandler: @escaping (URLRequest?) -> Void))
```

Session Data Delegate Methods

DataTask & UploadTask Events

```
func urlSession(URLSession, dataTask: URLSessionDataTask, didReceive: URLResponse,
  completionHandler: @escaping (URLSession.ResponseDisposition) -> Void)
// ResponseDisposition: cancel, allow, becomeDownload
func urlSession(URLSession, dataTask: URLSessionDataTask,
  didBecome: URLSessionDownloadTask)
func urlSession(URLSession, dataTask: URLSessionDataTask, didReceive: Data)
func urlSession(URLSession, dataTask: URLSessionDataTask,
  willCacheResponse: CachedURLResponse,
  completionHandler: @escaping (CachedURLResponse?) -> Void)
```

Session Download Delegate Methods

Download Task Events

```
func urlSession(URLSession, downloadTask: URLSessionDownloadTask,
  didWriteData: Int64, totalBytesWritten: Int64,
  totalBytesExpectedToWrite: Int64)
func urlSession(URLSession, downloadTask: URLSessionDownloadTask,
  didFinishDownloadingTo: URL)
```

Demo

Challenge Time!

```swift
let urlString = "http://localhost:3000/posts/"
```

```swift
func urlSession(_ session: URLSession,
  dataTask: URLSessionDataTask,
  didReceive response: URLResponse,
  completionHandler: @escaping (URLSession.ResponseDisposition) -> Void)
```

```swift
func urlSession(_ session: URLSession,
  dataTask: URLSessionDataTask,
  didReceive data: Data)
```