

# **Effective Networking**

## **with Swift and iOS 8**

# Ben Scheirman

**@subdigital**

# ChaiOne





**NS** *Screencast*

# Agenda

- Old and Crusty NSURLConnection
- New Hotness
- **Live** Demos 🙌
- HTTP
- Caching
- Bonus Round: API Tips

## NSURLConnection

# NSURLConnection

Invented for Safari ~2000

Made public in 2003

Poor separation of settings, config, cache





Back



Forward



Stop



Refresh



Home



AutoFill



Print



Mail

Address: <http://www.google.se/>

go

[Microsoft Web Sites](#) [MSN Web Sites](#) [Apple](#)[Web](#) [Images](#) [Videos](#) [Maps](#) [Translate](#) [Scholar](#) [Gmail](#) [more »](#)[Settings](#) | [Sign in](#)[Advanced Search](#)  
[Language Tools](#)

Google Search

I'm Feeling Lucky

Google.se offered in: [svenska](#)[Advertising Programs](#) - [Business Solutions](#) - [About Google](#) - [Go to Google.com](#)©2010 - [Privacy](#)

**NSURLConnection** was replaced by **NSURLSession** in iOS 7

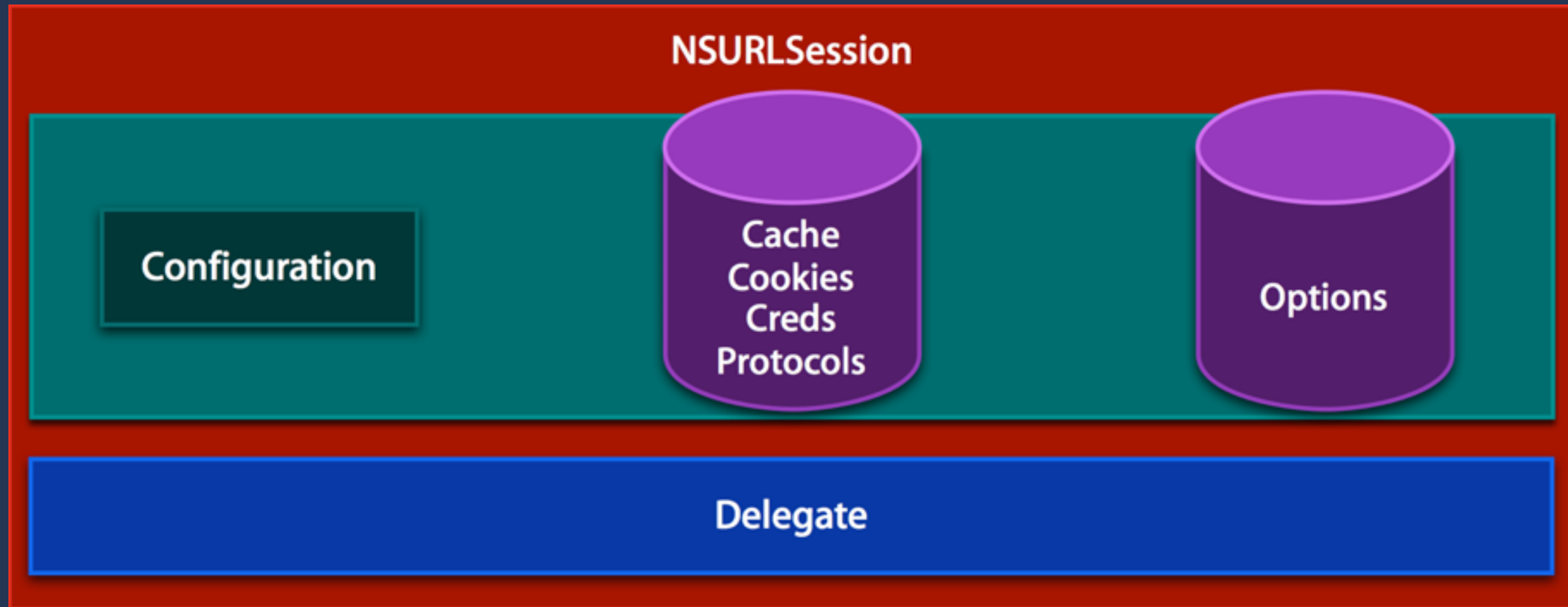
**NSURLSession**

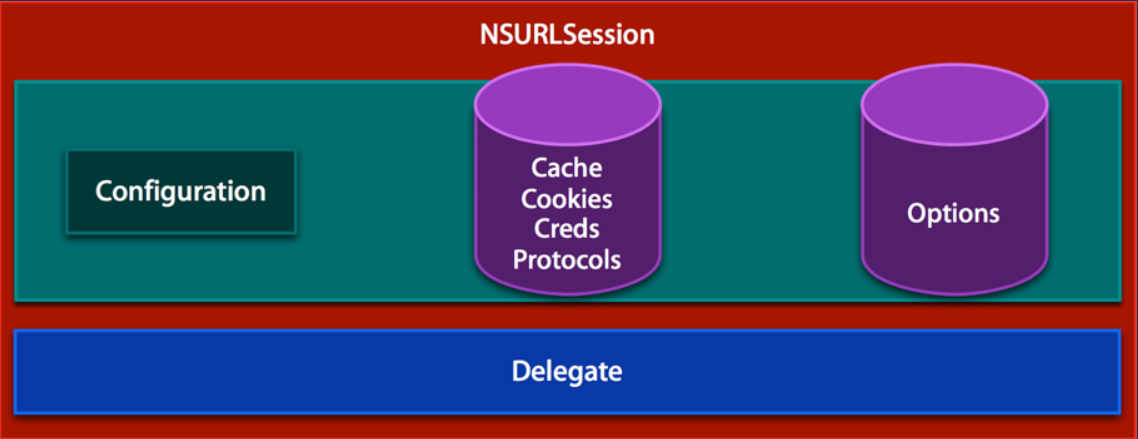
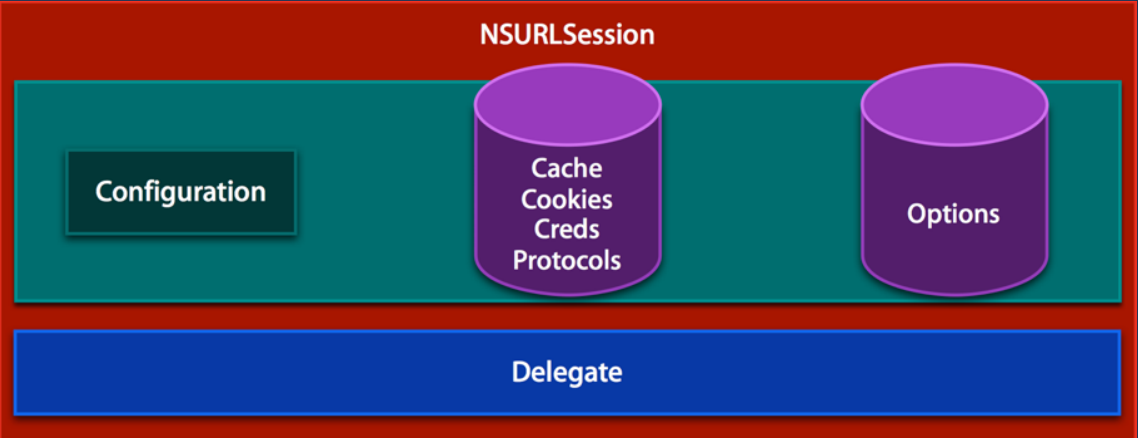
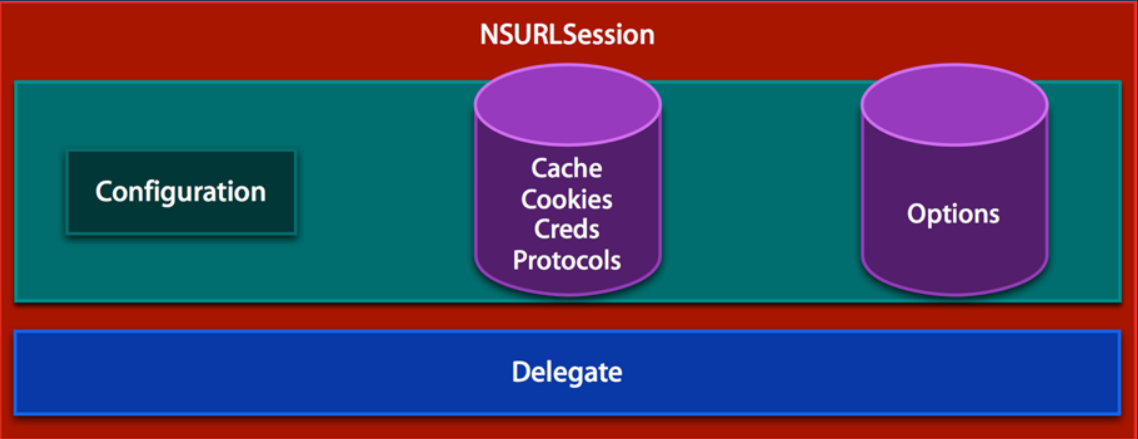
Still use NSURLRequest

Configurable Container (isolation w/ 3rd party libraries)

Improved auth

Rich Callbacks





# **The New Family**

NSURLSessionConfiguration

NSURLSession

NSURLSessionTask

NSURLSessionDelegate

# **Start with** NSURLSessionConfiguration

# Default Sessions

+ defaultSessionConfiguration

# Ephemeral Sessions

+ ephemeralSessionConfiguration

# Ephemeral Sessions

+ ephemeralSessionConfiguration

?

**"private browsing"**



# Background Sessions

+ `backgroundSessionConfiguration:`

# Background Sessions

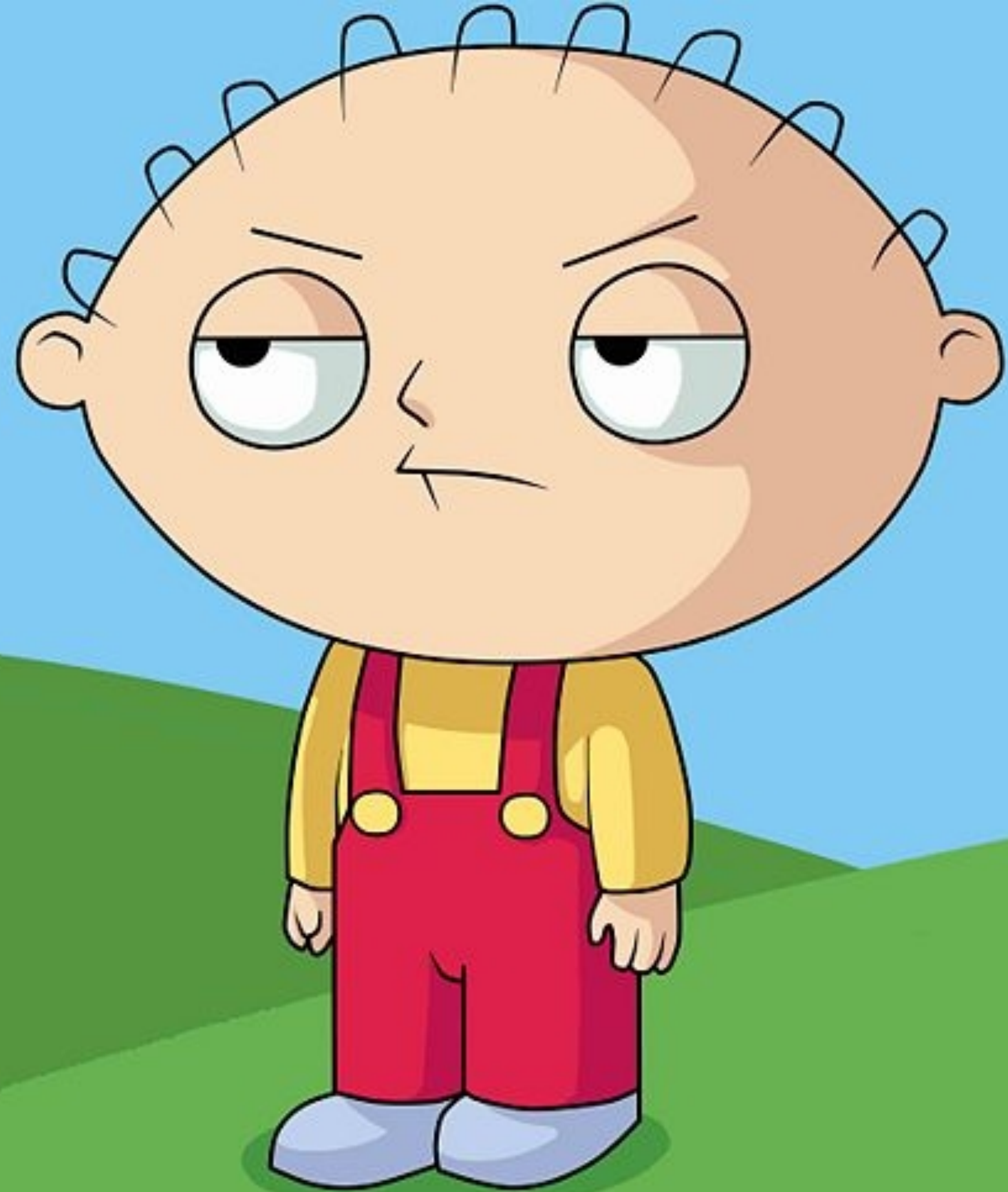
+ `backgroundSessionConfiguration:`

- Takes an `NSString` identifier
- Causes uploads / downloads to occur in background

# Customize it further

All of these class methods return a copy, tweak from there...

Well, what can you  
do with it?



# Set a default header

```
var config = NSURLSessionConfiguration.defaultSessionConfiguration()  
  
config.HTTPAdditionalHeaders = [ "auth_token" : "1234abcd" ]
```

# Mark Requests as low-priority

```
config.discretionary = true
```

- *Note: This has benefits, such as retrying when connection terminates, avoiding request if user is low on battery, or if Wi-Fi performance is not good enough.*

# Disable Cellular

```
config.allowsCellular = false
```

# Set custom caching behavior

```
config.URLCache = MyCustomCache()  
config.requestCachePolicy = NSURLRequestReturnCacheDataElseLoad
```

**Inject your own custom  
protocols!**

ben : // awww . yeah

(actually quite useful for  
mocking requests)

Next Step: build an  
**NSURLSession** to make  
requests

```
var session = NSURLSession(configuration: config)
```

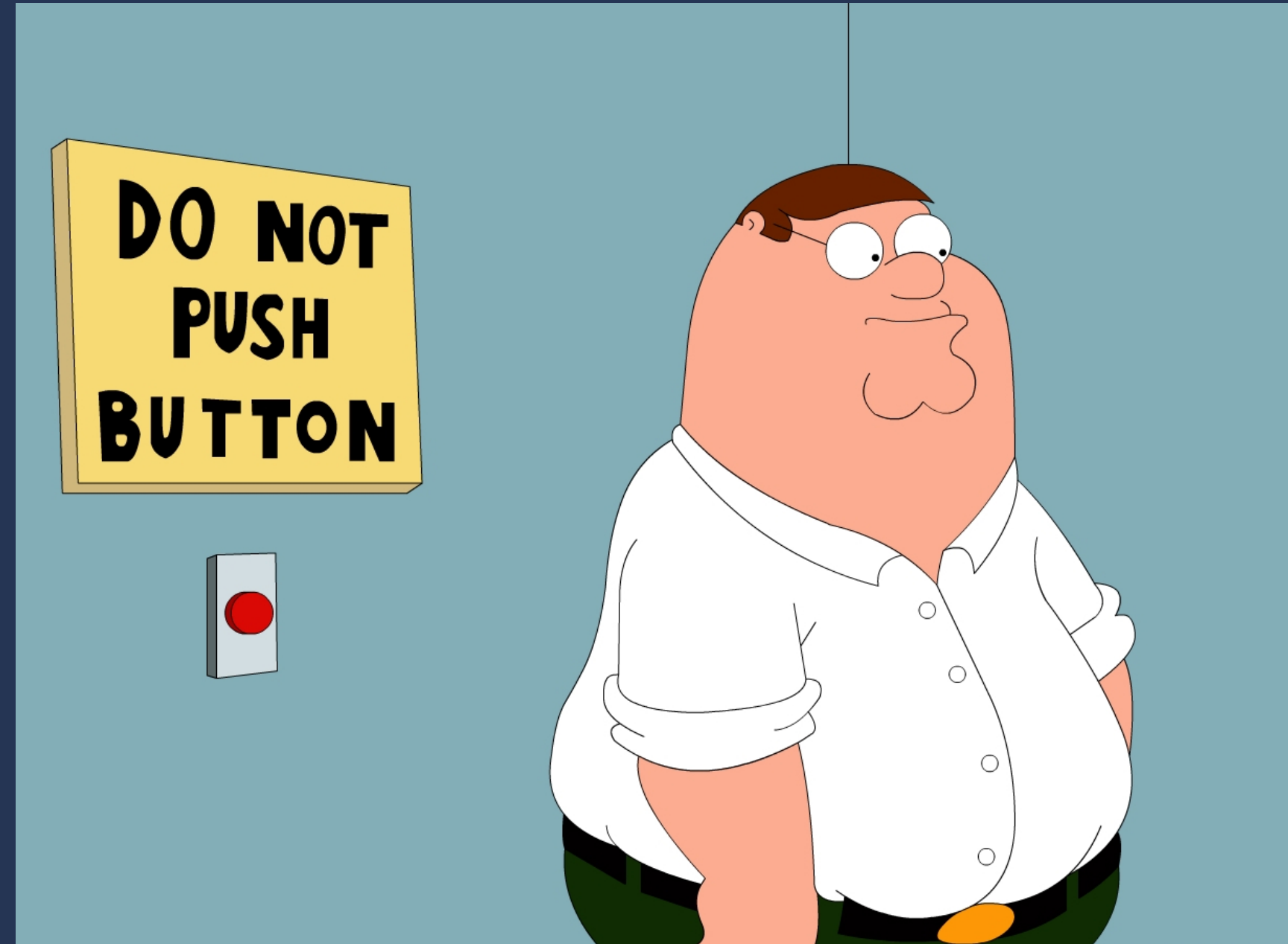
```
var session = NSURLSession(configuration: config)
```

**or with a delegate...**

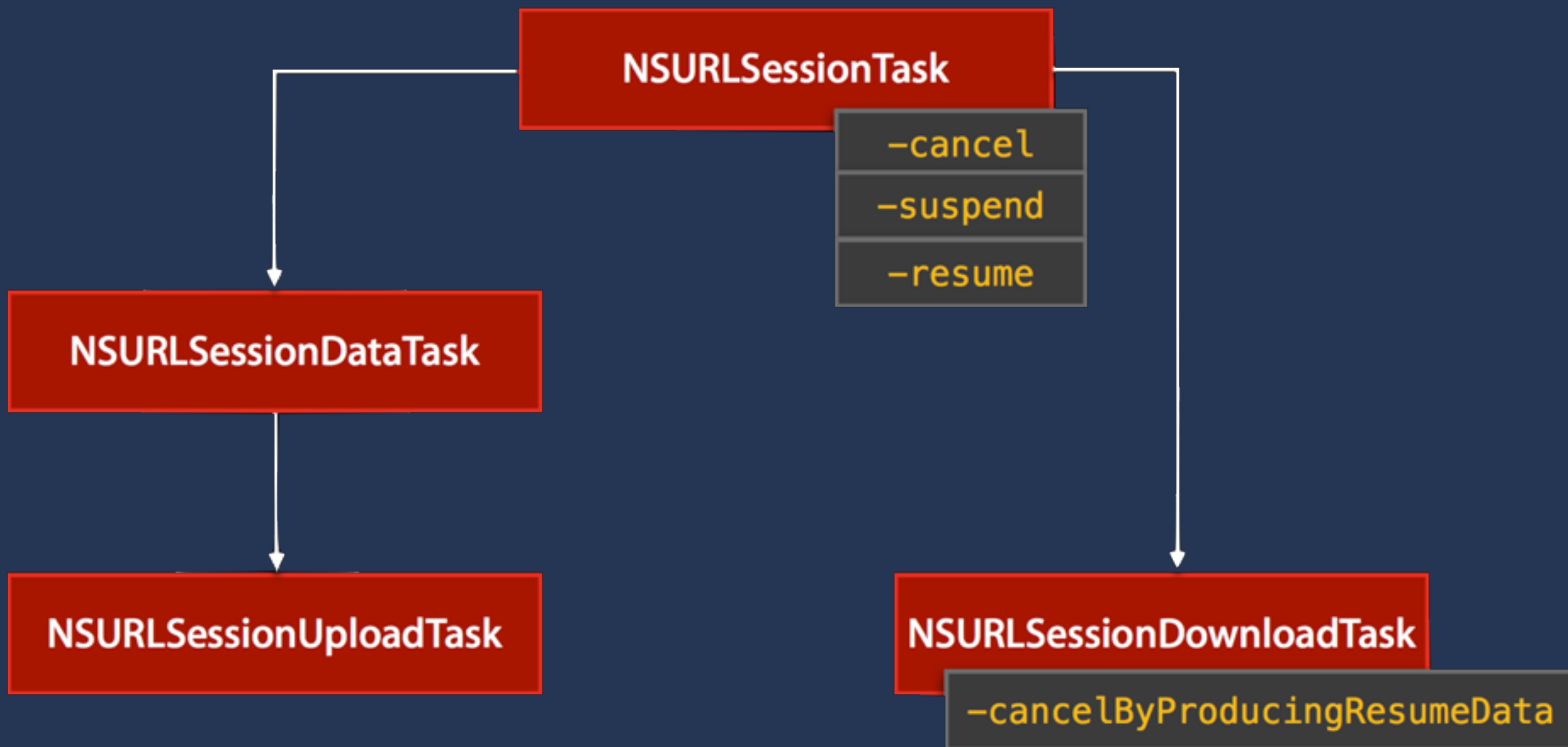
```
var delegate: NSURLSessionDelegate = self
var session = NSURLSession(configuration: config,
    delegate: delegate,
    delegateQueue: NSOperationQueue.mainQueue())
```

# NSURLSession

- Construct without the delegate if you want to use block callbacks
- Construct with a delegate for advanced control (or background sessions)
- If you pass completion handler blocks, delegate methods are **not** called



**NSURLSessionTask**



# NSURLSessionDataTask

- Represents GET, PUT, POST, DELETE
- Handle Completion Callback
- If error is present, request failed

# What Constitutes an Error?

- Connection failed
- Timeouts
- Host invalid
- Bad URL
- Too many redirects
- ... dozens more (check URL Loading System Error Codes)

# NSURLSessionDownloadTask

- Downloading a file / resource
- Streams to disk
- Useful when size is large and can't fit in memory
- Temp file path is provided in completion block
- **MUST** move it somewhere if you want it to stick around

# Building a simple GET Request

- Build an instance of `NSURLSessionDataTask`
- (optionally) give it a block callback\*
- Call `resume`

```
var config = NSURLSessionConfiguration.defaultSessionConfiguration()  
var session = NSURLSession(configuration: config)
```

```
var config = NSURLSessionConfiguration.defaultSessionConfiguration()  
var session = NSURLSession(configuration: config)  
  
var url = NSURL(string: "https://www.nsscreencast.com/api/episodes.json")
```

```
var config = NSURLSessionConfiguration.defaultSessionConfiguration()
var session = NSURLSession(configuration: config)

var url = NSURL(string: "https://www.nsscreencast.com/api/episodes.json")

var task = session.dataTaskWithURL(url) {
    (let data, let response, let error) in
    // ...
}
```

```
var config = NSURLSessionConfiguration.defaultSessionConfiguration()
var session = NSURLSession(configuration: config)

var url = NSURL(string: "https://www.nsscreencast.com/api/episodes.json")

var task = session.dataTaskWithURL(url) {
    (let data, let response, let error) in
    // ...
}

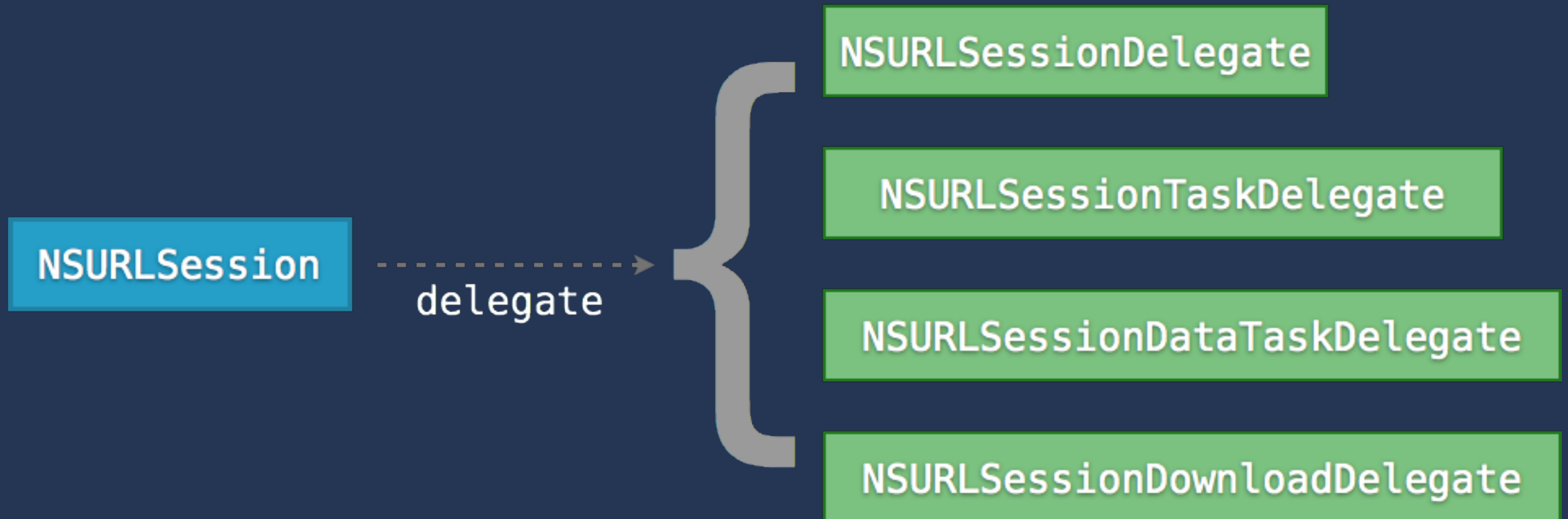
// don't forget to trigger the request
task.resume()
```

# NSURLSessionDelegate

Provide a delegate if you need more advanced control over:

- Download Progress
- Authentication Challenges
- Connection Failure

# NSURLSessionDelegate



# **Case Study: Requesting Images**

# Have you ever seen this?

```
NSURL *imageUrl = [NSURL URLWithString:@"http://i.imgur.com/kwpjYwQ.jpg"];  
  
NSData *imageData = [NSData dataWithContentsOfURL:imageUrl];  
  
UIImage *image = [UIImage imageData:imageData];  
  
[imageView setImage:image];
```



# What is wrong here?

```
NSURL *imageUrl = [NSURL URLWithString:@"http://i.imgur.com/kwpjYwQ.jpg"];  
  
NSData *imageData = [NSData dataWithContentsOfURL:imageUrl];  
  
UIImage *image = [UIImage imageData:imageData];  
  
[imageView setImage:image];
```

# These are blocking calls

- `NSData *imageData = [NSData dataWithContentsOfURL:imageUrl];`
- `UIImage *image = [UIImage imageData:imageData];`

# Sister hits moose on way to visit sister who hit moose

**WRY. 43) One woman in 100 Mile House hospital, the other recovering in Vancouver**

BY LARRY WAGNER  
STAFF WRITER

A pair of Klondike shovels have laid to rest the lives of two women with one in the house.

Monday ago, Christine Bradley, 49, badly injured when she hit a moose with her vehicle, was in Vancouver recovering from her injuries.

Christine Bradley, 49, was in Vancouver recovering from her injuries.

But sister Christine Bradley, 38, faced the future — she also hit a moose and ended up in hospital.

At the start of the week, a moose was Bradley was on her way home from a business trip when a moose hit her car.

The impact went through the windshield and landed on her, leaving a broken wrist, arm and hand, five fractured ribs and bleeding in her brain.

The pregnant mother died near the collision.

The calf did not survive.

When Bradley came out of her

coma last night, Christine Bradley and her husband 50-year-old decided to visit her.

She and her husband were in two cars last Friday afternoon. In the first car, she was going around a corner near 70 Mile House when she saw "a brown bear."

"That's right, it was a moose," she said. "I slammed on the brakes with both my feet."

"It was like two explosions. [The moose] came through the windshield on the passenger side, and then the body flipped around and the shoulder came into the side window."

She was taken to the hospital at 100 Mile House with nearly 50 wounds.

The collision of falling a moose just like her sister "could have been a trap."

"My first thought was, 'Are the moose going out [in a] hunting season for my family?'" Bradley said. "So far, we win three because we got three of them dead."

Moose are involved in about eight per cent of all wildlife vehicle collisions, according to the Wildlife Collision Prevention Program's website.

"Moose will attempt to avoid vehi-

cles by turning into a ditch," said Jeff Knight, spokesman for B.C.'s Ministry of Environment.

"It's a little bit of a trap," he said. "If you're on the road and the moose is on the road."

Bradley was released from hospital Saturday, and Monday she was home, where she said she was better.

"You can never tell when a moose or some other wildlife will be in front of you," she said. "I've been almost dead from this. I don't know how people who have died."

By making the program

## Can on tailgating may be lifted

**Demo:**

Downloading Images

- Images should probably be requested with download tasks
- Block-based callbacks can be unwieldy (especially in Objective-C)
- Use the delegate if you want to report download progress

**Demo:**

Searching the iTunes Store



# Status Code Cheat Sheet

1xx - Informational / Transient

2xx - A-OK 👍

3xx - It's over there 📍

4xx - You Messed up 🙅

5xx - We Messed Up 💣

# HTTP Caching

# HTTP Caching Techniques

- If-Modified-Since
- If-None-Match
- Cache-Control

# If-Modified-Since

- Client requests a resource
- Server responds, includes a Date response header
- Client Caches the data, including the date
- Client sends request header If-Modified-Since
- Server compares, can return 304 (Not Modified) with no body

## If-None-Match

- Client requests a resource
- Server responds, includes a E-Tag header
- Client Caches the data, including the E-Tag
- Client sends request header I f-None-Match
- Server compares, can return 304 (Not Modified) with no body

# Cache-Control

- Server indicates when the resource expires
- Client caches the data until that time
- Client will immediately return local cache data if still fresh



Client

Server

```
GET /customer/152.json HTTP 1.1  
Host: example.com
```



```
HTTP/1.1 200 OK  
Content-Type: application/json  
Content-Length: 37142  
Etag: "e6482391249374127ca94128"  
Cache-Control: max-age=60, public  
Date: Sun, 21 Oct 2012 16:15:37
```

Client

Server

```
GET /customer/152.json HTTP 1.1  
Host: example.com  
If-None-Match: "e6482391249374127ca94128"
```



```
HTTP/1.1 304 Not Modified
```

```
< no response body >
```



Client

Server

```
GET /customer/152.json HTTP 1.1  
Host: example.com  
If-Modified-Since: <date>
```



```
HTTP/1.1 304 Not Modified
```

```
< no response body >
```





```
def show
  @band = Band.find(params[:id])
  fresh_when(:etag => @band,
             :last_modified => @band,
             :public => true)
  expires_in 10.minutes, :public => true
end
```

# Observations

- Server still executes a query to compute E-Tag and Modified Date
- No body is transferred for a matching E-Tag or Date
- Client doesn't even make request if Cache-Control is used and content is still fresh

# Cache-Control sounds perfect

- Not everything is inherently cacheable
- Only helpful for a single client, *after* the initial request

Server



Cache-Control: max-age: 600



Cache-Control: max-age: 600

# **Reverse-Proxy Cache to the Rescue**

**Server**



**Varnish / Squid / Nginx**

Cache-Control: max-age: 600



Cache-Control: max-age: 600



Cache-Control: max-age: 600

# Pros / Cons of Cache-Control

- Client doesn't wait for a network request
- Server spends zero resources
- Clients May Render Stale Data

# Pros / Cons of E-Tag & IMS

- Server skips rendering
- Miniscule Data Transfer
- Can appear instantaneous\*
- Server still spending resources computing & comparing E-Tag & dates

# Reason About Your Data

- List of US States
- User's Profile
- Customer's Order
- Activity Timeline
- Yesterday's stats

Tradeoff between  
**fresh data** and **user experience**

# Caching with `NSURLSession`

- Uses `NSURLCache` out of the box
- Customize on the `NSURLSessionConfiguration` instance
- Use `ephemeralSessionConfiguration` to disable caching.

# NSURLCache **Gotchas**

# NSURLCache **Gotchas**

- will **not** cache on disk if max-age is less than 5 minutes

# NSURLCache **Gotchas**

- will **not** cache on disk if max-age is less than 5 minutes
- might not cache at all if **Cache-Control** isn't passed

# NSURLCache **Gotchas**

- will **not** cache on disk if max-age is less than 5 minutes
  - might not cache at all if **Cache-Control** isn't passed
  - might choose an arbitrarily large max-age if none provided
- \*

# NSURLCache **Gotchas**

- will **not** cache on disk if max-age is less than 5 minutes
- might not cache at all if **Cache-Control** isn't passed
- might choose an arbitrarily large max-age if none provided  
\*
- might not cache if size > 5% of available capacity

# Tuning the built-in cache

```
let MB = 1024 * 1024
var cache = NSURLConnection(memoryCapacity: 10 * MB,
                             diskCapacity: 50 * MB,
                             diskPath: nil)

sessionConfiguration.URLCache = cache
```

# Default Cache Location

- `~/Library/Caches/com.acme.app/Cache.db`

Navicat Essentials for SQLite - trial version 30 days left. [Buy Now!](#)

Connection Table View Index Trigger Query

Cache

- main
  - Tables
    - cfurl\_cache\_blob\_data
    - cfurl\_cache\_receiver\_data
    - cfurl\_cache\_response**
    - cfurl\_cache\_schema\_version
    - sqlite\_sequence
  - Views
  - Indexes
  - Triggers
  - Queries

cfurl\_cache\_response @main (Cache)

Image Text Hex Import Wizard Export Wizard Sort Ascending Sort Descending Remove Sorting

| entry_ID | version | hash_value  | storage_policy | request_key        | time_stamp      |
|----------|---------|-------------|----------------|--------------------|-----------------|
| 1        | 0       | 1424005679  | 0              | https://alpha-api. | 2012-10-19 17:2 |
| 2        | 0       | -1305208724 | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 3        | 0       | 1572491858  | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 4        | 0       | -1997901260 | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 5        | 0       | -777154780  | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 6        | 0       | 66153744    | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 7        | 0       | -404209947  | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 8        | 0       | -1931015091 | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 9        | 0       | 338276037   | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 10       | 0       | 1772610311  | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 11       | 0       | -769696739  | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 12       | 0       | -1381255225 | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 13       | 0       | 2140559844  | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 14       | 0       | -358207116  | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 15       | 0       | -1622839865 | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 16       | 0       | 1750647411  | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 17       | 0       | 352022890   | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 18       | 0       | -1578321287 | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 19       | 0       | 1302241355  | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |
| 20       | 0       | 1535007898  | 0              | https://d2rfichhc2 | 2012-10-19 17:3 |

select \*, rowid "NAVICAT\_ROWID" from "main"."cfurl\_cache\_response"

219 records in page 1

What do I have to do?

# Maybe nothing!

- Server should return appropriate cache headers
- Tweak caching behavior in `willCacheResponse` delegate method

# **The Content Flicker Problem**

# The Content Flicker Problem

- Data is already cached and fresh with E-Tag / IMS info
- Client *must* validate content is still fresh and wait for a 304
- Response is fast, but not fast to avoid drawing empty screen
- ...flicker

# **Resolving the Content Flicker Problem**

1. Return cached data immediately (if we have it)
2. Proceed with Request
3. Update callback (again) with fresh data

```
let url = NSURL(string: "http://cache-tester.herokuapp.com/contacts.json")  
let request = NSURLRequest(URL: url)
```

```
var task = session.dataTaskWithRequest(request)  
task.resume()
```

```
let url = NSURL(string: "http://cache-tester.herokuapp.com/contacts.json")
let request = NSURLRequest(URL: url)

if let cachedResponse = config.URLCache.cachedResponseForRequest(request) {
    // update UI
    processData(cachedResponse.data)
}

var task = session.dataTaskWithRequest(request)
task.resume()
```

# New in iOS 8: getCacheResponseForTask:

- Provides asynchronous cache fetch:

```
let url = NSURL(string: "http://localhost:3000/contacts.json")
let request = NSURLRequest(URL: url)
```

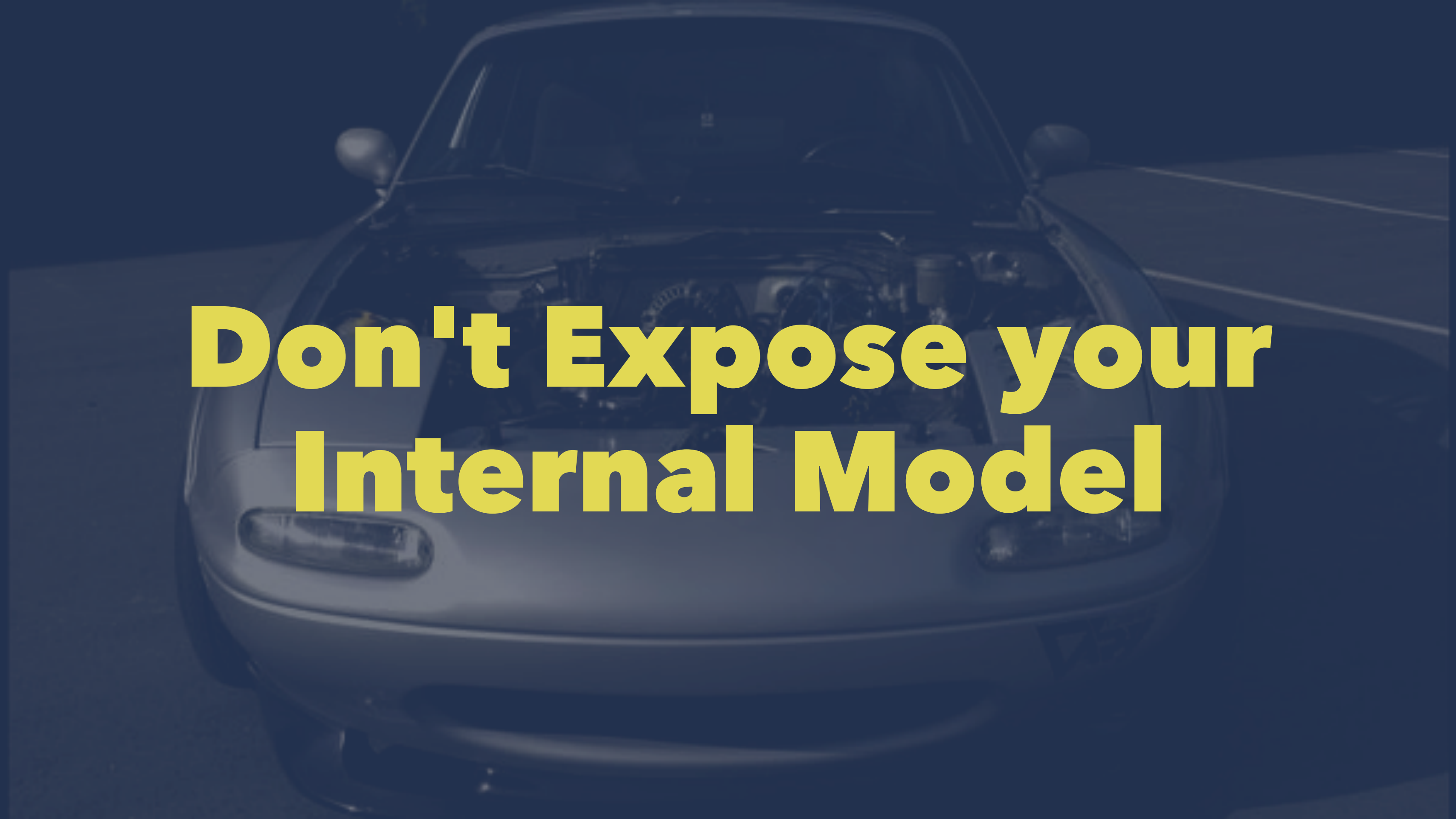
```
var task = session.dataTaskWithRequest(request)
```

```
config.URLCache.getCacheResponseForDataTask(task) {
    (let cachedResponse: NSCachedURLResponse?) in
        if cachedResponse != nil {
            self.processData(cachedResponse!.data)
        }

    task.resume()
}
```

# Bonus Round

API Tips

A dark blue-tinted photograph of the front of a classic car, likely a Lotus Evita, with its hood open, revealing the engine. The car is centered in the frame, and the background is dark and out of focus.

**Don't Expose your  
Internal Model**

A blue-tinted photograph of a railway bridge with tracks receding into the distance. The bridge has a complex steel truss structure. The tracks lead towards a large arched opening in the bridge. The background shows a hilly landscape with some vegetation.

# **Version Your API**

http://logrhythm.com/

GET / HTTP/1.1

Host: logrhythm.com

User-Agent: Mozilla/5.0 (X11; Linux i686 on x86\_64; rv:10.0.2) Gecko/20100101 Firefox/10.0.2

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gzip, deflate

Cookie: .ASPXANONYMOUS=BLAH.....;

WRUID=1243607642

DNT: 1

Connection: keep-alive

HTTP/1.1 200 OK

Date: Mon, 23 Apr 2012 20:55:58 GMT

Server: Microsoft-IIS/6.0

X-Powered-By: PleskWin, ASP.NET

X-Powered-By-Plesk: PleskWin

X-AspNet-Version: 2.0.50727

Set-Cookie: ViewMobile=False; path=/; HttpOnly

Set-Cookie: language=en-US; path=/; HttpOnly

Cache-Control: private

Content-Type: text/html; charset=utf-8

Content-Length: 88701

# Send Device Info as Headers



**Turn on Gzip  
Compression**

The background features a dark blue-grey gradient with a faint, semi-transparent line graph. The graph has a vertical axis on the left with labels '40ms' and '50ms' and a horizontal axis at the bottom with labels '12:00' and '18:00'. A dark blue line fluctuates across the graph, representing data points over time.

# Measure Response Times

# **Page Unbounded Data Sets**

# Thank You!

ben@scheirman.com

@subdigital • @nsscreencast

benscheirman.com • nsscreencast.com

**speakerdeck.com/subdigital/  
ios8-networking**