Asynchronität

Callbacks, Threading und Operationen mit dem iPhone SDK

Problemstellung

- Daten aus dem Internet laden (Bilder, XML / JSON, …)
- Langsame Ladevorgänge mit Mobilverbindung (EDGE, 3G)
- Latenz


Carrier
9:07 AM
Tweets
iphone_e...
RIM Introduces BlackBerry Storm2 to Fix Flaws, Challenge IPhone http://bit.ly/2ukzi1
YourTech...
RotorBlog - 9 Free iPhone Apps to Get You into the Halloween Mood http://bit.ly/1gVA9X
Andriavega
Yay just learned how to control my iPod while in on the Internet oh iPhone I sometimes love u... I'll love u even more...
mediraa
Iphone knp si ih dr td lemot banget
hrkznd
横のねーちゃんもiphoneでtwitterしてます。香港って凄い！
krystynch...
@MzPinkee oh yeah, I got my iphone. see what tweets are being left out of my timeline. does that happen to you?

Synchrones Laden

- Einfachste Lösung
- initWithContentsOfURL:
- Ausführung in RunLoop des Main Thread

Synchrones Laden

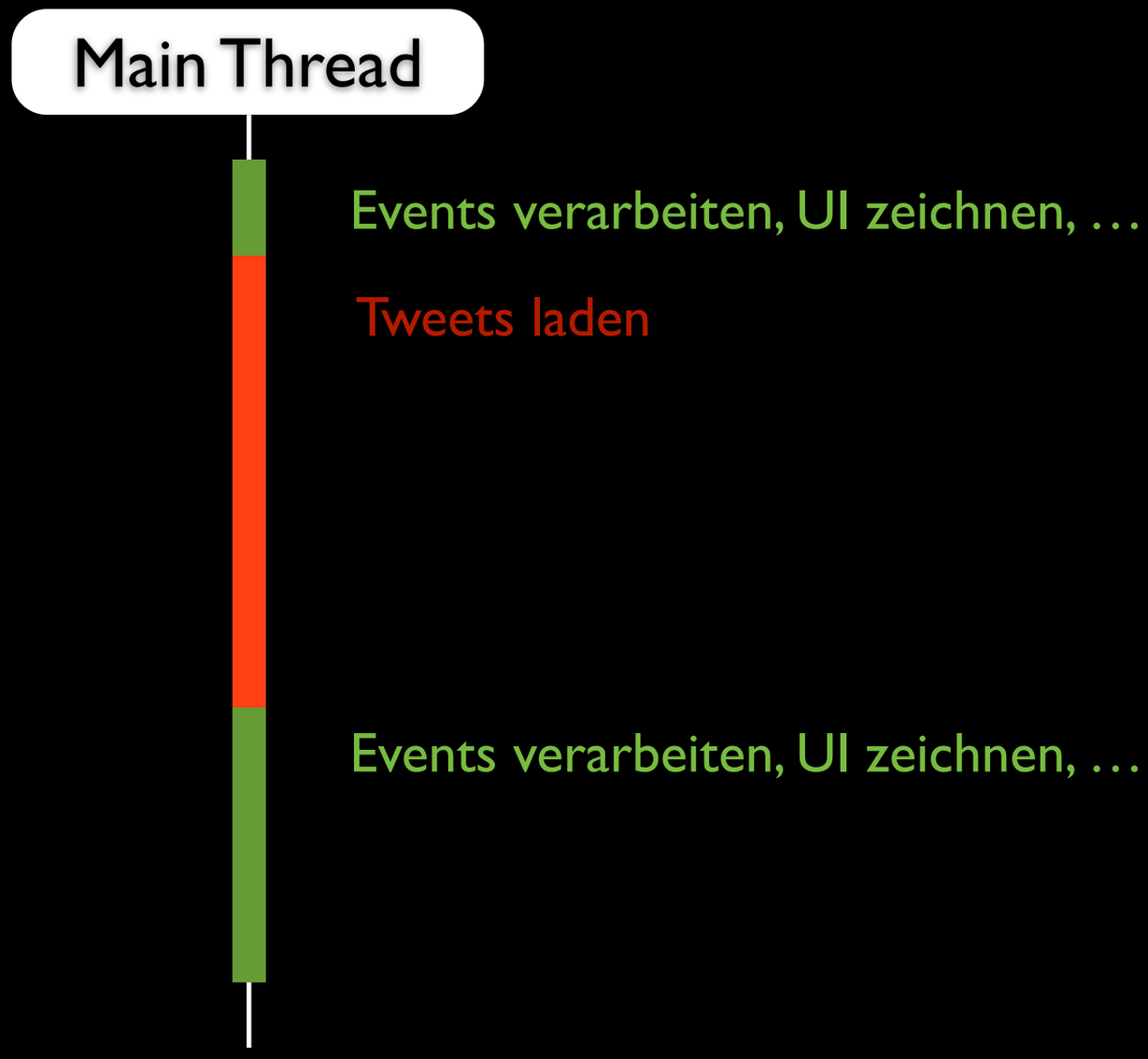
PRO

- sehr einfach

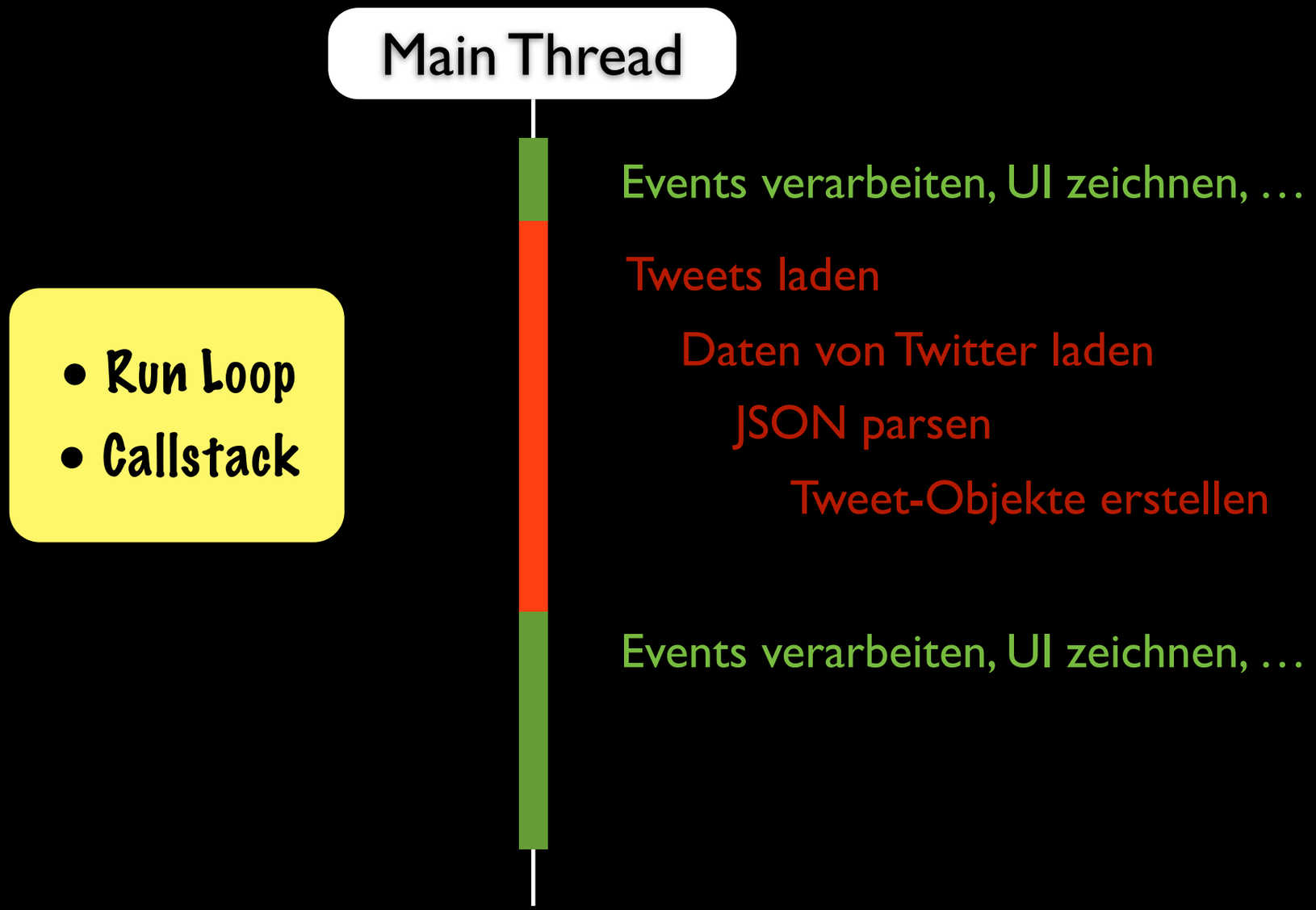
CON

- UI blockiert

Synchrones Laden



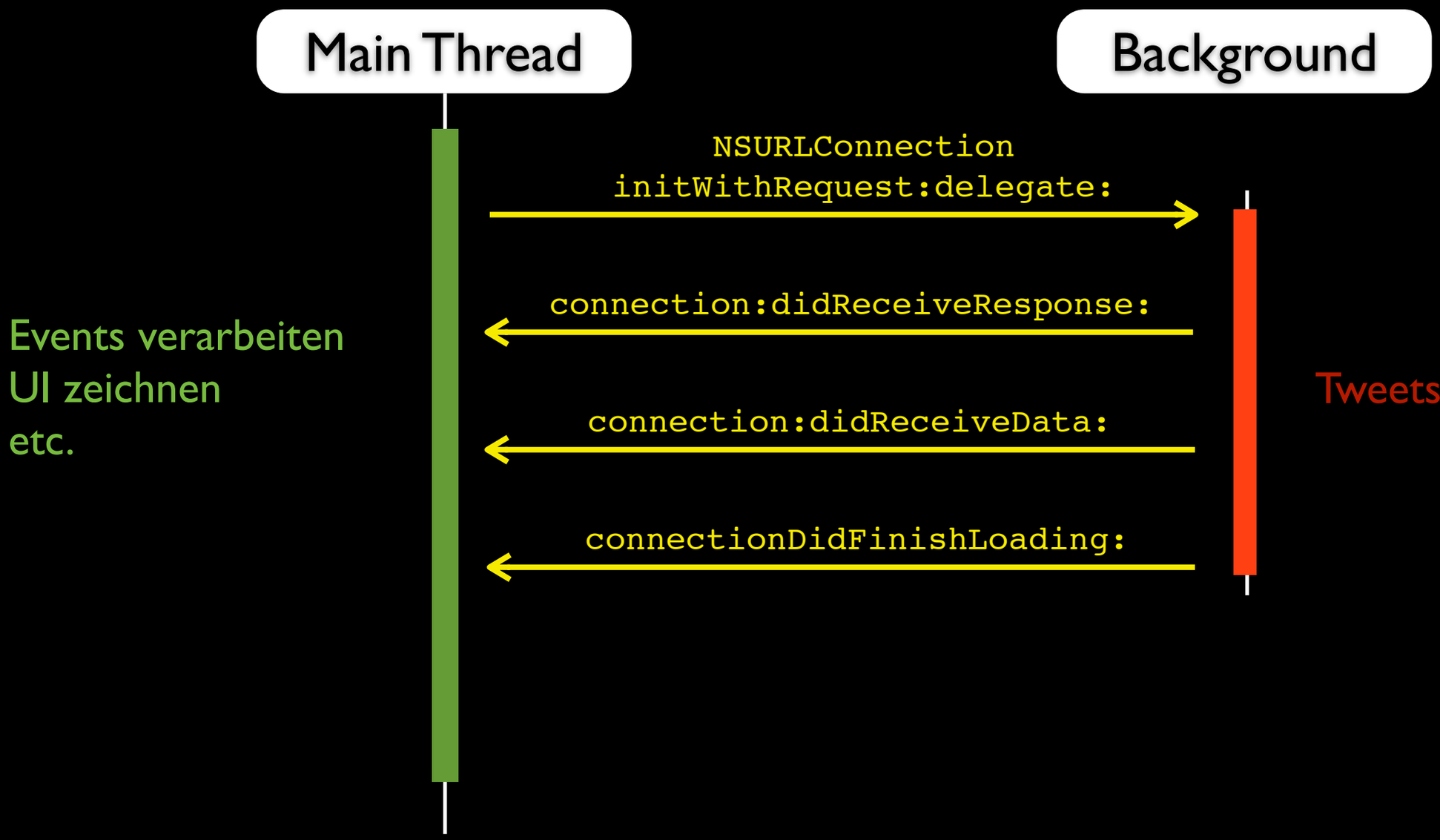
Synchrones Laden



Callback APIs

- Delegation Methods
- connection:didReceiveData:
- Für viele Klassen verfügbar
- Abarbeitung in Hintergrund-Thread
- Callbacks im RunLoop des Main Thread

Callback APIs



Callback APIs

PRO

- Einfach
- In vielen Klassen verfügbar
- Aktionen lassen sich abbrechen
- Error Handling

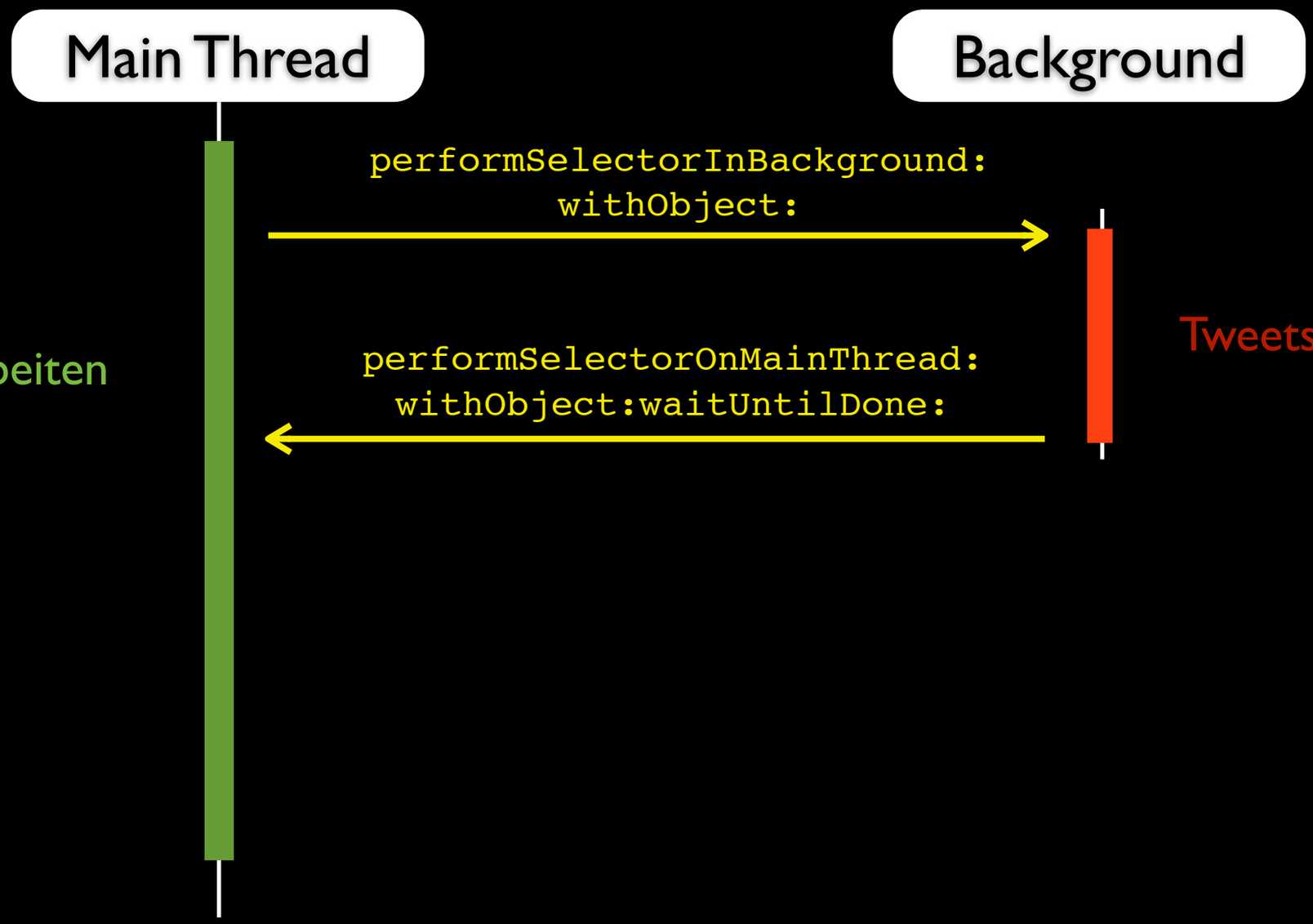
CON

- Relativ viel Code
- Daten verwalten

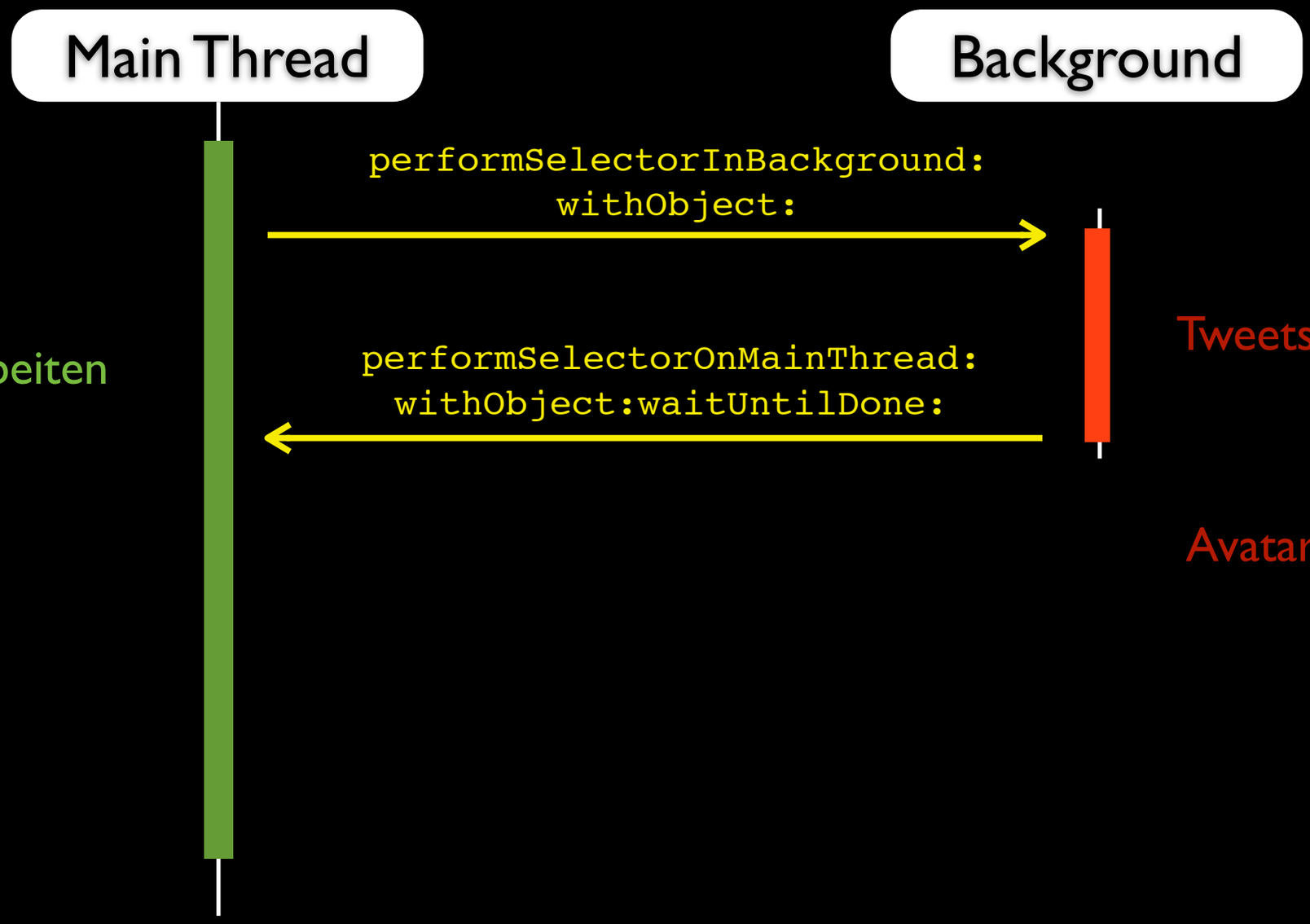
Threading

- Direktes Erzeugen eines neuen Threads
- performSelectorInBackground:
- Eigener Autorelease-Pool
- Callback muss auf den Main Thread!

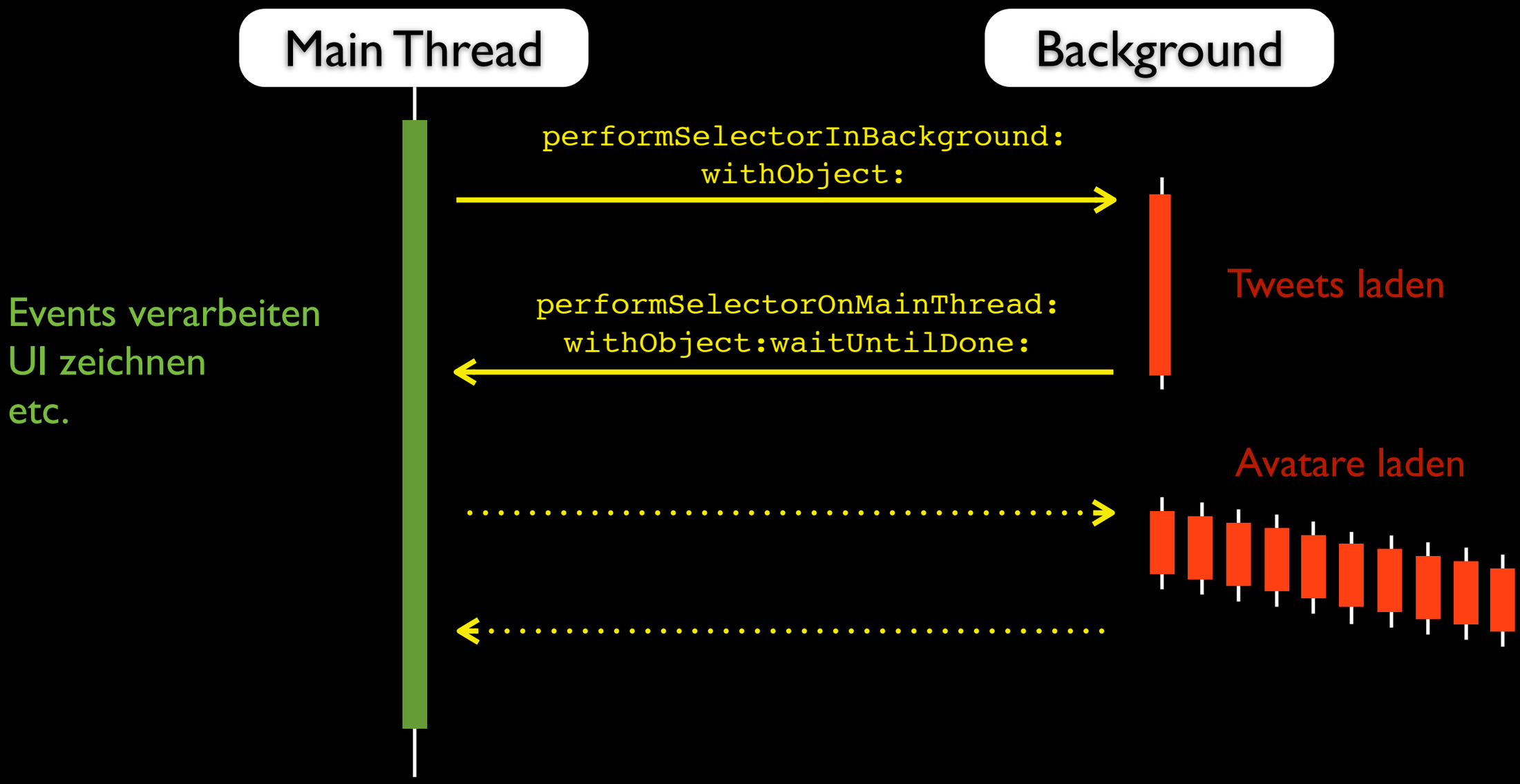
Threading


Main Thread
Background
performSelectorInBackground:
withObject:
performSelectorOnMainThread:
withObject:waitUntilDone:
Events verarbeiten
UI zeichnen
etc.
Tweets laden

Threading


Main Thread
Background
performSelectorInBackground:
withObject:
performSelectorOnMainThread:
withObject:waitUntilDone:
Events verarbeiten
UI zeichnen
etc.
Tweets laden
Avatare laden

Threading


Main Thread
Background
performSelectorInBackground:
withObject:
performSelectorOnMainThread:
withObject:waitUntilDone:
Events verarbeiten
UI zeichnen
etc.
Tweets laden
Avatare laden

Threading

PRO

- Kein Extracode

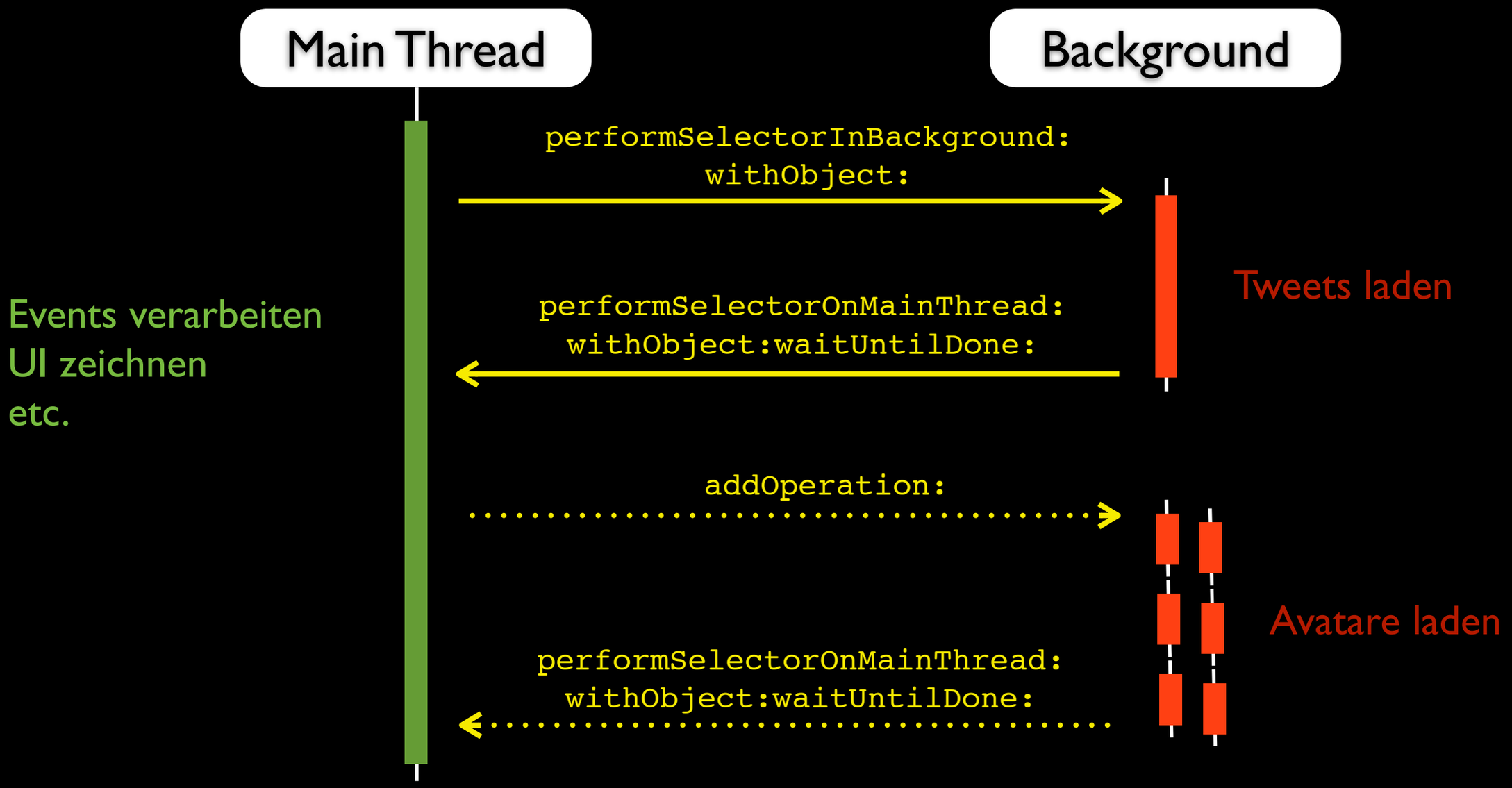
CON

- Mehrere Threads sind schwer zu verwalten
- Nicht der empfohlene Weg

Operationen

- NSOperation und NSOperationQueue
- Abarbeitung in separaten Threads
- Kein eigener Autorelease-Pool
- Callback muss auf den Main Thread!
- Anzahl Threads kann festgelegt werden
- Operationen lassen sich canceln

Operationen



Operationen

PRO

- Saubere Verwaltung mehrerer Threads
- Aktionen sind gekapselt und wiederverwendbar

CON

- Bereits gestartete Operationen lassen sich nicht abbrechen

Vergleich der Ansätze

	Abarbeitung im Hintergrund	Eigener Autorelease Pool	Eigene Threadverwaltung
Synchron	nein	-	-
Callback	ja	nein	nein
Threading	ja	ja	ja
Operations	ja	nein	nein

Fazit & Empfehlungen

- Einfache Anforderungen mit Delegation lösen (Error Handling leicht machbar)
- Operations verwenden , sobald mehrere Ressourcen nachgeladen werden müssen
- Verschiedene OperationQueues für unterschiedliche Controller/Bereiche
- Operations lassen sich auch canceln

Links

- Beispiel-Code auf GitHub
- Tutorial: NSOperation and NSOperationQueue
- Tutorial: Managing Concurrency with NSOperation
- iPhone Dev Center: Threading Programming Guide

Vielen Dank :)

Dennis Blöte
http://dennisbloete.de
Twitter: @dbloete