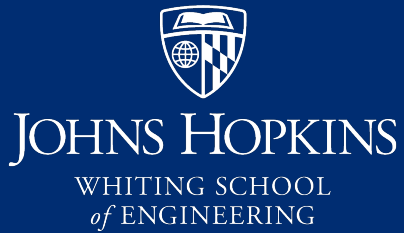
Johns Hopkins Engineering

User Experience

Module 3

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING



Asynchronous Multithreading

Lots of cores == Lots of Threads! (Right?)

- Modern day processors have multiple cores
- The more cores, the more threads (or programs)
- Most of these programs are small (e.g. short running)
- What's needed is efficient ways to manage multiple threads
- Threaded code doesn't scale well with arbitrary # of cores!

Grand Central Dispatch

- Instead of relying on threads, Apple uses an asynchronous design approach to deal with concurrency
- Asynchronous operations start a task (on a thread) and return to the caller. When the task is complete the caller is notified
- Grand Central Dispatch handles this thread creation, management and communication for you, in a nice developer friendly API!

Dispatch Queues

- Serially or Concurrently
- FIFO
- Simple API, Automatic thread pool management
- Fast, Memory efficient
- Scale!

http://www.appcoda.com/grand-central-dispatch/

Work Items

- Dispatch queues work on blocks of code called Work Items
  - Can be written along with the queue creation
  - Can also get assigned to the queue and reused

- Execution can be:
  - Synchronous: app doesn’t exit code block until it completes
  - Asynchronous: work item is invoked, and control is returned to the app immediately

http://www.appcoda.com/grand-central-dispatch/

DispatchWorkItem

- In addition to passing in code via blocks or closures, you can also encapsulate that code in a DispatchWorkItem

```
1  func useWorkItem() {
2      var value = 10
3      let workItem = DispatchWorkItem {
4          value += 5
5      }
6  }
7
8  workItem.perform()
9  //OR, to put on another queue
10 let queue = DispatchQueue.global()
   queue.async(execute: workItem)
```


http://www.appcoda.com/grand-central-dispatch

Avoiding the Main Queue

- UI rendering and interaction occurs on the Main Queue
- Long running tasks run on background threads
  - But these threads shouldn't try to change the UI!
- UI tasks can be dispatched back the Main Queue for processing

http://www.appcoda.com/grand-central-dispatch

NSOperationQueue

- OperationQueues are the equivalent of a concurrent dispatch queue
- While DispatchQueues are FIFOs, OperationQueues allow dependencies to be established between tasks
- These tasks are encapsulated in NSOperations

NSOperation

- NSOperations contain tasks that are submitted to NSOperationQueues
- NSOperation is abstract, so you need to define custom classes to perform your tasks

Priority

- Operations can have a priority
  - veryLow
  - low
  - normal
  - high
  - veryHigh
- Priority can impact the placement in the queue, hence impacting when it gets executed

http://nshipster.com/nsoperation/

Quality of Service (QoS)

- QoS helps tell the OS how much CPU, network and disk resources are given to your task
  - userInteractive – handling UI events or drawing
  - userInitiated - user requested, immediate request for data
  - utility – user or system requested, not immediately needed
  - background – not user initiated or invisible
  - default – no QoS information provided

http://nshipster.com/nsoperation/

Dependency

- OperationQueues can order thread execution based on dependencies amongst the Operations
- This comes in handy when you want to break up your tasks into smaller chunks, each of which has a clear dependency (this task must finish before the next one starts)
- `operation2.addDependecy(operation1)` states that operation 1 must be completed before operation 2 can begin

http://nshipster.com/nsoperation/

CompletionBlocks

- Upon an Operation's completion, it will execute its completionBlock once. You can define it like this:

- ```
  let operation = Operation()
  operation.completionBlock = { print("Completed") }
  ```


http://nshipster.com/nsoperation/
  ```

Which one to use?

- If you're doing a one-time calculation, or trying to speed up an existing method, use a lightweight GCD dispatch call rather than NSOperation
- If your tasks have dependencies, priorities, require a specific quality of service, or need to have the option of being cancelled, use an NSOperation
- They aren't mutually exclusive – you may use both in your app!

http://nshipster.com/nsoperation/