Multithreading in iOS

Threading is an important concept in iOS. The concept is pretty simple. This is what happens inside the processor. Consider launching an app in your iPhone. When the app launches, it will be on the main thread or the UI thread. At this point, when we try to do some time consuming task in the main thread, the UI will stop responding for a while. This is a situation the user will never want to face. From the users perspective, the app should always be responding and should be fast. As we know , most of the modern processors supports multitasking and are pretty fast. So, instead of doing the time consuming task in the main thread, better give it to different thread so that the main thread can do the other work it has to perform.

Grand Central Dispatch or **GCD** is a low-level API for managing concurrent operations. It will make your application smooth and more responsive. Also helps for improving application performance. Sometimes we are trying to perform multiple tasks at the same time that time most of the developer-facing application hang or freezing issue this is the common issue. That's why we are using GCD to manage multiple tasks at the same time.

Most of the time you probably used this code frequently:

```
DispatchQueue.main.async {
    // Perform your async code here
}
```

DispatchQueue
The DispatchQueue API like a company. Who having staff units like junior level and senior level workers. So now the company can take both heavy and light work with there team.

Concurrenct
It's starting multiple tasks at the same time but not guarantee for the finish at same time. Its can finish any order.

Serial
It's executing one task at a time.

Sync vs Async
Sync -When a work item is executed synchronously with the sync method, the program waits until execution finishes before the method call returns.

```
func syncWork(){
        let northZone = DispatchQueue(label:
    "perform_task_with_team_north")
        let southZone = DispatchQueue(label:
    "perform_task_with_team_south")

        northZone.sync {
            for numer in 1...3{ print("North \(numer)")}
        }
        southZone.sync {
            for numer in 1...3{ print("South \(numer)") }
        }
    }

    //Call Func here
```

```
    syncWork()

//Output
//    North 1
//    North 2
//    North 3
//    South 1
//    South 2
//    South 3
```

Async -execute asynchronously with the async method, the method call returns immediately.

```
func asyncWork(){
        let northZone = DispatchQueue(label:
"perform_task_with_team_north")
        let southZone = DispatchQueue(label:
"perform_task_with_team_south")

        northZone.async {
            for numer in 1...3{ print("North \(numer)") }
        }
        southZone.async {
            for numer in 1...3{ print("South \(numer)") }
        }
    }

//Call Async Task
asyncWork()


//OutPut
//    North 1
//    South 1
//    North 2
//    South 2
```

```
//    North 3
//    South 3
```

Global Queue -Using to perform non-UI work in the background.
Main Queue -Using to update the UI after completing work in a task on a concurrent queue.

DispatchQueue with delay:

```
let deadlineTime = DispatchTime.now() + .seconds(1)
DispatchQueue.main.asyncAfter(deadline:
deadlineTime) {
    //Perform code here
}
```