**Блоки отправки**

**Определение**

Блок отправки представляет собой кусок кода, который нужно выполнить асинхронно.

В Swift блок отправки представлен замыканием (closure) или функцией без имени.

**Создание блока отправки**

Блок отправки может быть создан в любом месте кода, как обычное замыкание или функция без имени.

В GCD блоки отправки используются для асинхронного выполнения кода в очередях диспетчеризации.

**Добавление блока отправки в очередь**

Блок отправки добавляется в очередь диспетчеризации с помощью метода async, sync или других методов, зависящих от типа очереди.

При добавлении блока отправки в очередь он выполняется асинхронно в отдельном потоке.

**Асинхронное выполнение**

Блок отправки выполняется асинхронно, что означает, что выполнение продолжается без ожидания завершения блока.

Это позволяет выполнять длительные или потенциально блокирующие операции, не замедляя главный поток выполнения.

**Параллельное выполнение**

Блоки отправки могут выполняться параллельно в нескольких потоках, если используются параллельные очереди диспетчеризации (DispatchQueue.concurrent).

**Завершение блока отправки**

После выполнения блока отправки вызывается соответствующее завершающее действие, если оно указано.

В случае асинхронного выполнения, код, следующий после добавления блока в очередь, продолжает выполняться независимо от выполнения блока.

**Применение блоков отправки**

Блоки отправки широко используются для асинхронного выполнения задач, таких как загрузка данных из сети, обработка изображений, выполнение долгих вычислений и других асинхронных операций.

**Пример**

Создание и выполнение DispatchWorkItem

1. // Создание DispatchWorkItem с помощью замыкания
2. let workItem = DispatchWorkItem {
3. // Этот код будет выполнен при выполнении workItem
4. print("Задача выполнена")
5. }
6. // Создание очереди диспетчеризации
7. let queue = DispatchQueue(label: "com.example.queue")
8. // Добавление workItem в очередь для выполнения
9. queue.async(execute: workItem)

**Пример**

Использование параметров flags и QoS

1. // Создание DispatchWorkItem с помощью замыкания и параметров flags и QoS
2. let workItem = DispatchWorkItem(qos: .userInitiated, flags: .detached) {
3. // Этот код будет выполнен при выполнении workItem
4. print("Задача выполнена")
5. }
6. // Создание очереди диспетчеризации
7. let queue = DispatchQueue(label: "com.example.queue")
8. // Добавление workItem в очередь для выполнения
9. queue.async(execute: workItem)