Cheatsheet – Networking/JSON-Parsing/Multithreading

Networking

URL(): Erstellung einer URL-Instanz aus dem String

```
let url = URL(string: "http://api.icndb.com/jokes")!
```

URLSession.shared: Erstellung einer gemeinsamen anwendungsweiten Session

```
let urlSession = URLSession.shared
```

URLSession.shared.dataTask(): Erstellung einer Aufgabe (Task), die den Inhalt der URL abruft und nach Abschluss einen Handler (completionHandler) aufruft

```
let task = urlSession.dataTask(with: url) { (data, response, error) in

}
```

task.resume(): Um die Aufgabe (den Task) zu starten

```
task.resume()
```

JSON-Parsing

Codable: Codierung von benutzerdefinierten Datenformaten wie JSON in native Swif-Objekte

```
struct Person: Codable {
  var name: String
  var age: Int
  var address: String
}
```

JSONDecoder().decode(): Gibt einen Wert des angegebenen Typs zurück, der aus einem JSON-Objekt dekodiert wurde.

```
do {
    let model = try JSONDecoder().decode(Person.self, from: jsonData)
    print(model)
} catch {
    print("Error parsing JSON")
}
```

Mutlithreading

Main Queue: Funktionen innerhalb der Main Queue werden im Hauptthread ausgeführt und haben die höchste Priorität

```
DispatchQueue.main.async{
  //Do any UI updates here
}
```

Global Queue: Funktionen innerhalb von Global Queue werden je nach Quality of Service mit einer bestimmten Priorität ausgeführt

```
DispatchQueue.global(qos: .background).async{
  //Do any heavy operation here
}
```

Kombination: Verteilung der Aufgaben

```
DispatchQueue.global(qos: .background).async{
  //Do any heavy operation here
  let image = downloadImageFromServer()
  DispatchQueue.main.async {
    self.imageView.image = image
  }
}
```