Swift Cheatsheet

Basic Structure and Variables

Swift is a modern, fast, and safe programming language by Apple for iOS, macOS, watchOS, and tvOS development.

- Declaration:
  - `let`: Constant (read-only).
  - `var`: Variable (mutable).
- Type Inference: `let x = 10` (Int).
- Explicit Type: `var message: String = "Hello"`.
- Basic Types: `Int`, `Double`, `Float`, `Bool`, `String`.

Control Flow

- `if-else`:

```
if count > 0 { ... } else { ... }
```

- `switch` (Requires exhaustive cases):

```
switch status {
case .ok: print("Success")
case .error(let msg): print("Error:
\(msg)")
default: break
}
```

- Loops:
  - `for item in collection { ... }`
  - `for i in 0..<5 { ... }` (Half-open range)
  - `while condition { ... }`

Optionals

Safety feature to handle missing values.

- `Int?`: Can be an Int or `nil`.
- Unwrapping:
  - `if let value = optionalValue { ... }`: Optional binding.
  - `guard let value = optionalValue else { return }`: Early exit.
  - `optionalValue ?? defaultValue`: Nil-coalescing operator.
  - `optionalValue!`: Force unwrapping (dangerous).

Functions and Closures

- Function:

```
func greet(person: String) → String {
    return "Hello, \(person)!"
}
```

- Closure: Self-contained blocks of functionality.

```
let sortedColors = colors.sorted { $0
< $1 }
```

Collections

- Array: `var names = ["Alice", "Bob"]`.
- Dictionary: `var ages = ["Alice": 25, "Bob": 30]`.
- Set: Unordered collection of unique values.

Structs and Classes

- Struct: Value type (copied on assignment).

```
struct Point {
    var x: Double
    var y: Double
}
```

- Class: Reference type (shared). Supports inheritance.

```
class Person {
    var name: String
    init(name: String) { self.name =
name }
}
```

Enums and Protocols

- Enum:

```
enum Direction {
    case north, south, east, west
}
```

- Protocol: Defines a blueprint of methods/properties.

```
protocol Drivable {
    func drive()
}
```

Error Handling

```
do {
    try someThrowingFunction()
} catch {
    print("Error: \(error)")
}
```

Modern Swift (SwiftUI & Concurrency)

- SwiftUI: Declarative UI framework.
- Async / Await: Modern concurrency.

```
func fetchData() async throws → Data
{ ... }
```

Google Style Guide

Core Principles

- Clarity at the point of use.
- Prefer `let` over `var` whenever possible.
- Use Swift's safety features (Optionals, Guard, Errors) instead of crash or silent failure.

Naming Conventions

- Classes, Structs, Enums, Protocols: `UpperCamelCase`.
- Variables, Methods, Enum cases: `lowerCamelCase`.
- Use descriptive names; avoid abbreviations (e.g., `width` instead of `w`).

Formatting

- Indentation: 2 spaces.
- Line Length: Max 100 characters.
- Braces: Trailing braces on the same line.
- Colon: No space before, one space after. `var x: Int`.

Programming Practices

- Forbid redundant `self` (use only when necessary for disambiguation).
- Prefer `struct` over `class` where reference semantics are not required.
- Access Control: Use `private` and `fileprivate` to restrict visibility.
- Documentation: Use `///` or `/** ... */` for documentation comments.

Comments

- No unnecessary comments that restate the code. Explain complex architectural decisions or "why".