Welcome to Swift Meet up

Swift Ahemdabad @ CricHeroes

- Self Taught Developer
- 2x WWDC Winner

- Technical writer and speaker
- Worked on apps in multiple category
- Currently SDE2 @ Upstox (Mumbai)

Swift Excellence
Enhancing code quality

Kanishka Chaudhry @ Swift Ahemdabad

Diving in “Why?” And not just “What’s recommended?”


Any UI inconsistency problem in iOS
iOS developer
DispatchQueue.main.async{}

Difference between “Getting Job Done” and “Excellent” Dev Is

1. Attention to Detail
2. Optimisation
3. Curiosity and the pursuit of “Why?”

1. isEmpty > count == 0

Count iterates through the array.

isEmpty stops checking as soon as its realises its non empty

```swift
extension Array {
    subscript(safe index: Int) -> Element? {
        return indices.contains(index) ? self[index] : nil
    }
}
```

Usage:

```swift
overlayLabels[safe: indexPath.row]
```

2. Count check for arrays

So your app doesnt crash like your portfolio 💥

3.
array.first > array[0]
array.last > array[count]

Safer + Less Iteration

4. endIndex > count

5. Skipping explicit return if only one statement

Less Verbose (just like gen z short forms 😎)

```
func getCellHeight(for indexPath: IndexPath) { cellHeight }
```

6. Mutability - constant and private(set) variables

Reduces scope of mutation.

7. Avoid explicit self

The lesser the verbose, the better

Bonus: Don’t take PR comments personally

Remember the number of comments on your first PR?

8. Guard > if let

Readability.

Indentation.

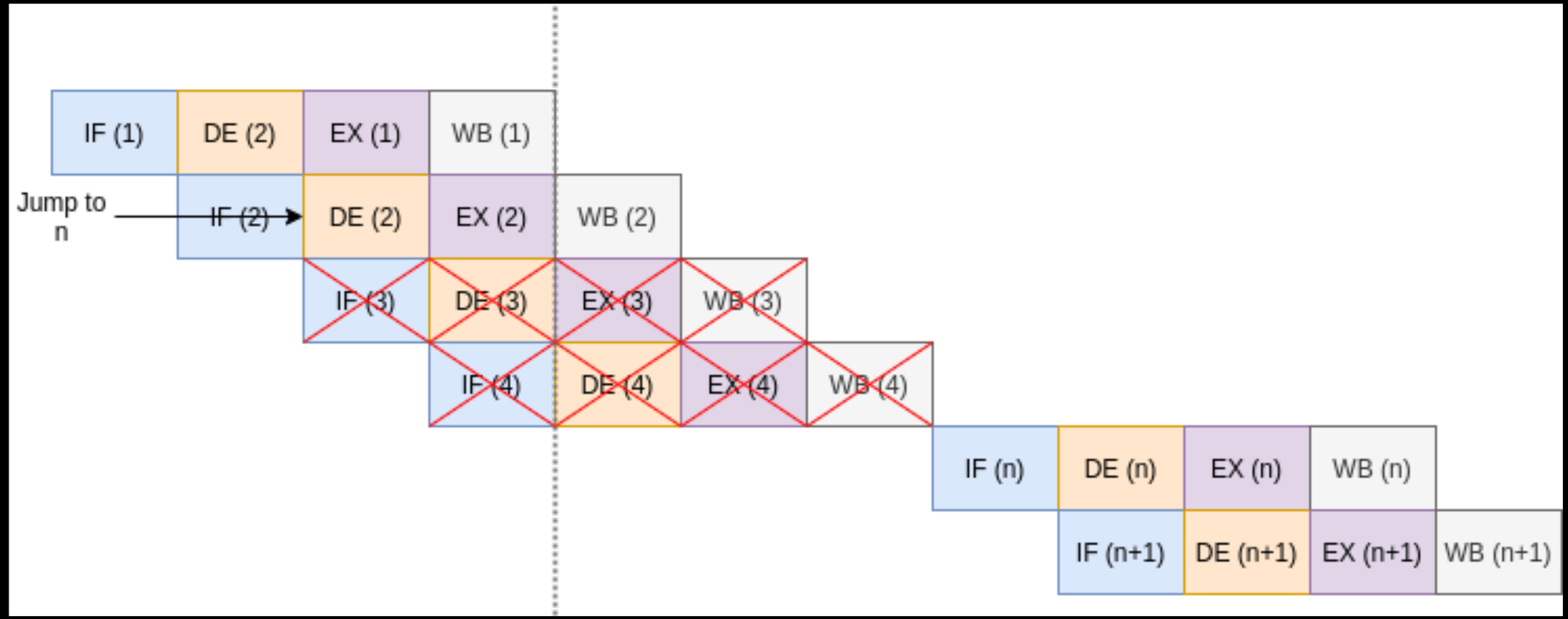
Less blocks of code.

Branchless Code (more on this soon)

9. Ternary Operator Wherever Possible

```
let title = overlayLabels.isEmpty ? "No Overlays" : "\(overlayLabels.count)"
```

Bonus: Branchless Programming



10. Of course, weak reference in delegate and escaping closures !

Closures can be as pushing coding without running if not inspected 🧐

Experiment: Bonus: Try an experiment of checking number of VCs in memory

LeakSanitzers can be used for identifying memory leaks

11. Relevant comments where magic numbers are used

12. Checking for 0 in denominator when dividing numbers

Avoiding crashes just in case - specially when you code is server driven

13. `@objc optional` for optional protocol methods if possible

Keep in mind, this requires your objects to be Objective C driven

```swift
protocol NetworkRequestDelegate {
    func didStartRequest(url: URL)
    func didFinishRequest(url: URL, success: Bool)
    func didFailRequest(url: URL, error: Error)
}

extension NetworkRequestDelegate {
    func didStartRequest(url: URL) {}
    func didFinishRequest(url: URL, success: Bool) {}
    func didFailRequest(url: URL, error: Error) {}
}
```

```swift
@objc protocol NetworkRequestDelegate {
    @objc optional func didStartRequest(url: URL)
    @objc optional func didFinishRequest(url: URL, success: Bool)
    @objc optional func didFailRequest(url: URL, error: Error)
}
```

14. Make it *sufficiently* generic

Just enough. Examine the usual life cycle of the code in your organisation

15. Avoiding too many parameters in protocol

```
protocol VideoControllerHelperProtocol {
    func didTapPlay(videoPosition: Float,
                    videoItem: String,
                    muted: Bool,
                    isUserInitiated: Bool,
                    isFullscreen: Bool)
}
```

```swift
struct VideoControllerHelperParamBuilder {
    let videoPosition: Float
    let videoItem: String
    let muted: Bool
    let isUserInitiated: Bool
    let isFullscreen: Bool
}
protocol VideoControllerHelperProtocol {
    func didTapPlay(param: VideoControllerHelperParamBuilder)
}
```

16. String Interpolation > String Addition

```
let finalString = "Total number of items:" + " " + "\(count)" // ❌

let efficientString = "Total number of items: \(count)" // ✅
```

17. Explicit Type Declaration are slightly better than inferred type declaration

```
var x: Int = 10 // explicitly declared type

var y = 10 // inferred type
```

18. `final class` if no scope for inheritance

19. Avoid tight coupling using concrete types

Protocol for the win.

19.1 Put methods in extensions if not likely to be overridden

The End

Now, time for you to craft excellent code!