Swift Pills 💊 #1

Vincent Pradeilles - Worldline

Encapsulating `[weak self]`

Encapsulating [weak self]

```
handler = { [weak self] argument in
    guard let self = self else { return }

    // use `self`
}
```

Encapsulating [weak self]

```
handler = { [weak self] argument in
    guard let self = self else { return }

    // use `self`
}
```

Encapsulating [weak self]

```
protocol Weakifiable: class { }
```

Encapsulating [weak self]

```
protocol Weakifiable: class { }

extension NSObject: Weakifiable { }
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }

    func weakify<T>(_ code: @escaping (T, Self) -> Void) -> (T) -> Void {
        return { [weak self] arg in
            guard let self = self else { return }

            code(arg, self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }

    func weakify<T>(_ code: @escaping (T, Self) -> Void) -> (T) -> Void {
        return { [weak self] arg in
            guard let self = self else { return }

            code(arg, self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }

    func weakify<T>(_ code: @escaping (T, Self) -> Void) -> (T) -> Void {
        return { [weak self] arg in
            guard let self = self else { return }

            code(arg, self)
        }
    }
}
```

Encapsulating [weak self]

```
extension Weakifiable {
    func weakify(_ code: @escaping (Self) -> Void) -> () -> Void {
        return { [weak self] in
            guard let self = self else { return }

            code(self)
        }
    }

    func weakify<T>(_ code: @escaping (T, Self) -> Void) -> (T) -> Void {
        return { [weak self] arg in
            guard let self = self else { return }

            code(arg, self)
        }
    }
}
```

Encapsulating [weak self]

```
handler = { [weak self] argument in
    guard let self = self else { return }

    // use `self`
}
```

Encapsulating [weak self]

```
handler = weakify { strongSelf, argument in
    // use `strongSelf` 🎉
}
```