Coding Guideline v11

Par Vincent Ducastel 12 Ajoutez une réaction

Objectifs

L'objectif de cette guideline est d'avoir du code :

- plus lisible
- plus maintenable
- formaté de la même façon par toute l'équipe

Légende

TODO : Règle à écrire

OK : Validé par l'équipe (review conseillé)

KO : Non validé par l'équipe

NO RULE : Pas de règle trouvé (pas de review)

Liste

- On peut utiliser le “!” dans une condition courte, sinon on déclare une variable pour faire le “!” KO

- On peut exposer la valeur d’un Relay OK
- On définit une variable (à la place de $0) si il y a plusieurs closure OK
- Si une seule closure longue, on peut laisser $0 NO RULE
- Alignement des paramètres OK
- On privilégie la clarté des opérateurs plutôt que les paramètres KO

- Combine
  - Associe un publisher simplement à un subject NO RULE
- SwiftUI
- DRY (Don’t Repeat Yourself)
  - On définit une variable si plusieurs utilisations OK
- Autres
  - Abréviations NO RULE
  - Acronyme NO RULE
- Extension TODO
  - Une extension globale
  - Une extension par profession
  - On les range ou ?
    - Dans la v12
- Aérer le code
  - On saute une ligne avant et après le MARK: NO RULE
  - Sauter des lignes dans les méthodes NO RULE
  - Eviter de mettre de la logique dans les paramètres KO
  - Saut de ligne après la première accolade ? NO RULE

- TODO ?
- Idée ?

Catégories

Structure d’une classe

Regrouper les propriétés par visibilité OK

Regrouper les propriétés par visibilité, `private`, `internal`, `protected` etc.

Ajouter des `// MARK :`

```
// MARK: - Internal Properties
// MARK: - Private Properties
// MARK: - Life Cycle
// MARK: - Internal Functions
// MARK: - Private Functions
```

DO

```
internal class GoodService: BaseService {

    // MARK: - Internal properties

    internal let alertingConfiguration: Observab
    internal let isCarplayAvailable: Observable<

    // MARK: - Private properties

    private let alertingConfigurationRelay: Beha
    private let isCarplayAvailableRelay: Behavio
```

DON’T

```
class GoodService: BaseService {
    internal let alertingConfiguration: Observabl
    private let alertingConfigurationRelay: Behav

    internal let isCarplayAvailable: Observable<B
    private let isCarplayAvailableRelay: Behavior
}
```

```
12
13      // MARK: - Life cycle
```



```
16  }
```

Regrouper par protocol KO

Non validé par l'équipe, privilégier l'affichage des variables en début de classe

> Regrouper par protocol

Toujours spécifier `internal` OK

Toujours spécifier `internal` (même si `internal` est implicite en Swift)

v11

DO

```
1  internal class GoodService: BaseService {
2
3      // MARK: - Internal properties
4
5      internal let property1 = "VALUE_1"
6      internal let property2 = "VALUE_2"
7      internal let property3 = "VALUE_3"
8  }
```

DON'T

```
1  class GoodService: BaseService {
2
3      // MARK: - Internal properties
4
5      let property1 = "VALUE_1"
6      let property2 = "VALUE_2"
7      let property3 = "VALUE_3"
8  }
```

v12 ? @Vincent Ducastel **TODO**

struct, enum, static OK

On déclare les

- struct
- enum
- static

en début de classe avant les variables et les méthodes

DO

```
1  internal class GoodService: BaseService {
2
3      // MARK: - struct, enum, static...
4
5      internal enum Level {
6          case debug
7          case info
8          case error
9      }
10
11     // MARK: - Internal properties
12
13     // etc...
```

DON'T

```
1  class GoodService: BaseService {
2
3      // MARK: - Internal properties
4
5      // etc...
6
7      // MARK: - Private properties
8
9      // etc...
10
11     // MARK: - struct, enum, static...
12
13     internal enum Level {
```

```
14
15      // MARK: - Private properties
```

```
14          case debug
15          case info
```



```
18
19      // MARK: - Life cycle
20
21      // etc...
22  }
```

```
18
19      // MARK: - Life cycle
20
21      // etc...
22  }
```

private (set) OK

On évite d'exposer des variables directement (get, set), la plupart du temps, on a besoin que du (get)

DO

```
1  internal class GoodService: BaseService {
2
3      // MARK: - Internal properties
4
5      internal private(set) var monitorCoordinator:
6      internal private(set) var accountCoordinator:
7  }
```

DON'T

```
1  internal class GoodService: BaseService {
2
3      // MARK: - Internal properties
4
5      internal var monitorCoordinator: MonitorCoord
6      internal var accountCoordinator: AccountCoord
7  }
```

imports dans l'ordre alphabétique OK

On met les imports dans l'ordre alphabétique

DO

```
1  import A
2  import B
3  import C
```

DON'T

```
1  import C
2  import B
3  import A
```

Méthodes

Retour à la ligne dans les paramètres des méthodes OK

Pour améliorer la visibilité :

- à partir de 3 paramètres
- si la ligne est trop longue et qu'elle revient à la ligne automatiquement

Très subjectif et contextuel

Tout dépend de la longueur de la ligne, si la ligne est difficile à lire, on met des retours à le ligne

DO

DON'T

```
routingService.calculateRoute(origin: currentLoca
                              destination: destin
```

```
routingService.calculateRoute(origin: currentLoca
    // Implementation...
```

Coding Guideline v11 Mise à jour : mars 01, 2024 Modifier

```
}
```

Comment nommer une méthode NO RULE

Pas de règle évidente pour l'instant

DO

exemple 1 : **playAndRecord()**

```
1
func playAndRecord(with sound: Sound) {}
playAndRecord(with: sound)

2
func playAndRecord(sound: Sound) {}
playAndRecord(sound: sound)

3
func playAndRecordWith(_ sound: Sound) {}
playAndRecordWith(sound)

4
func playAndRecord(_ sound: Sound) {}
playAndRecord(sound)
```

exemple 2 : **updateManeuversList()**

```
1
func updateManeuversList(with routeProgress: Rou
updateManeuversList(with: routeProgress)

2
func updateManeuversList(routeProgress: RoutePro
updateManeuversList(routeProgress: routeProgress

3
func updateManeuversListWith(_ routeProgress: Ro
updateManeuversListWith(routeProgress)

4
func updateManeuversList(_ routeProgress: RouteP
updateManeuversList(routeProgress)
```

exemple 3 : updateLocation**IfNeeded()**

Par moment, c'est pas mal d'avoir un peu de contexte directement dans le nom du méthode

```
func updateLocationIfNeeded(location: CLLocation
    if location.isValid() {
      return
    }
    updateLocation(location: location)
}
```

DON'T

```
1
```

```
7
8  func updateLocation(location: CLLocation) {
```

Coding Guideline v11 Mise à jour : mars 01, 2024 Modifier

return / guard

Retour à la ligne si il y'a plusieurs conditions OK

DO

```
guard let theme = ThemeCenter.default.currentThem
      let title = currentProductButtonTitle,
      !title.isEmpty else {
    // Implementation...

    return
}
```

DON'T

```
guard let theme = ThemeCenter.default.currentThem
    // Implementation...
    return
}
```

Pas de `return` dans les transformations si une seule ligne OK

DO

```
let track = coordinatesList.compactMap { CLLocati
```

DON'T

```
let track = coordinatesList.compactMap {
    CLLocation($0)
}

let track = coordinatesList.compactMap {
    return CLLocation($0)
}
```

Mettre en évidence le `return` si le guard est long ou complexe KO

Non validé par l'équipe, pas de règle applicable pour l'instant

C'est plus du "bon sens" à avoir pour rendre le code plus lisible par d'autre personne

> Mettre en évidence le return si le guard est long ou complexe

Array

coordinates ou coordinatesList NO RULE

On peut ajouter "list" pour les tableaux ou les liste.

DO

```
private class GoodService: BaseService {
```

```
2
3     let coordinatesList: [String] = []
```

Coding Guideline v11 Mise à jour : mars 01, 2024 Modifier

```
1  private class GoodService: BaseService {
2
3      let coordinates: [String] = []
4  }
```

On initialise toujours un tableau NO RULE

DO (preferred)

```
1  private class GoodService: BaseService {
2
3      private let maneuversListRelay: [DisplayableM
4      private let maneuversListRelay: BehaviorRelay
5  }
```

DO

```
1  private class GoodService: BaseService {
2
3      private let maneuversListRelay: [DisplayableM
4      private let maneuversListRelay: BehaviorRelay
5  }
```

Enum

.none (default) en première position OK

DO

```
1  public enum FavoriteType: String {
2      case none <------------------
3      case home
4      case work
5      case other
6  }
```

DON'T

```
1  public enum FavoriteType: String {
2      case home
3      case none <--------------
4      case work
5      case other
6  }
```

True / False

On évite le `property == true/false` , on utilise le ! pour la négation OK

DO

```
1  guard isEmpty else { return }
```

```
1  guard !isEmpty else { return }
```

DON'T

```
1  guard isEmpty == true else {
2      return
3  }
```

```
1  guard isEmpty == false else {
2      return
3  }
```

On peut utiliser le “!” dans une condition courte, sinon on déclare une variable pour faire le “!” KO

On peut utiliser le “!” dans une condition courte, sinon on déclare une variable pour faire le “!”

RxSwift

Eviter d’exposer des Relay, mais plutôt des Observable OK

DO

```
// MARK: - internal properties

internal var guidingStateObservable: Observable<
    guidingStateRelay.asObservable()
}

internal var isGuidingObservable: Observable<Boo
    isGuidingRelay.asObservable()
}

// MARK: - private properties

private let guidingStateRelay: BehaviorRelay<Gui
private let isGuidingRelay: BehaviorRelay<Bool>
```

DON’T

```
// MARK: - internal properties

internal let guidingStateRelay: BehaviorRelay<Gui
internal let isGuidingRelay: BehaviorRelay<Bool>
```

On peut exposer la valeur d’un Relay OK

Toutefois, privilégié d’abord le **binding** plutôt que l’accès à la valeur d’un relay.

DO

```
// MARK: - internal properties

internal var isGuiding: Bool {
    isGuidingRelay.value
}

// MARK: - private properties

private let isGuidingRelay: BehaviorRelay<Bool> =
```

DON’T

```
internal let isGuidingRelay: BehaviorRelay<Bool>
```

On définit une variable (à la place de `$0`) si il y a plusieurs closure OK

DO

```
currentRouteCoordinatesListRelay
```

DON’T

```
currentRouteCoordinatesListRelay
```

```
    .subscribe(onNext: { [weak self] coordinates
        if let coordinatesList {
            } else {
                self?.clientAPI.updateGuidingTrack(n
            }
    }).disposed(by: bag)
```

```
    .subscribe(onNext: { [weak self] in
        if let coordinatesList = $0 { // $0 == [
            } else {
                self?.clientAPI.updateGuidingTrack(n
            }
    }).disposed(by: bag)
```

Coding Guideline v11 Mise à jour : mars 01, 2024 Modifier

Si une seule closure longue, on peut laisser `$0` NO RULE

DO (preferred)

With variable `configuration`

```
mapViewModel.mapCameraConfigurationDriver
    .distinctUntilChanged()
    .drive(onNext: { [weak self] mapCameraConfig
        guard let self else { return }

        self.mapViewManager?.updateCameraConfigu
        // Because updateCameraConfiguration doe
        self.viewModel.updateZoomLevel(mapCamera
        self.mapViewManager?.updateCustomLocatio
    }).disposed(by: bag)
```

DO

With `$0`

```
mapViewModel.mapCameraConfigurationDriver
    .distinctUntilChanged()
    .drive(onNext: { [weak self] in
        guard let self else { return }

        self.mapViewManager?.updateCameraConfigu
        // Because updateCameraConfiguration doe
        self.viewModel.updateZoomLevel($0.zoomLe
        self.mapViewManager?.updateCustomLocatio
    }).disposed(by: bag)
```

Alignement des paramètres OK

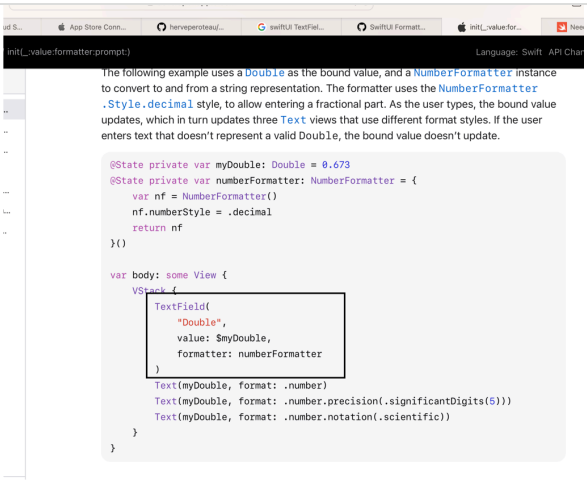
DO

```
Observable.merge(isDisplayingTrafficFlowObservab
                 isDisplayingBuildingsObservable
                 isDisplayingRoadNumbersObservab

Observable.merge(
    isDisplayingTrafficFlowObservable,
    isDisplayingBuildingsObservable,
    isDisplayingRoadNumbersObservable
)

Observable.merge(isTraffic, isBuilding, isRoadNu
```

DON'T

```
// Too long
Observable.merge(isDisplayingTrafficFlowObservabl
```

Coding Guideline v11 Mise à jour : mars 01, 2024 Modifier



On privilégie la clarté des opérateurs plutôt que les paramètres KO

Non validé par l'équipe, pas de règle applicable pour l'instant

> On privilégie la clarté des opérateurs plutôt que les paramètres

Combine

Associe un publisher simplement à un subject NO RULE

DO (preferred)

```
remoteDatabaseApiService?.synchronisationStatusPu
    .subscribe(synchronisationStatusSubject)
    .store(in: &subscriptions)
```

DO

```
remoteDatabaseApiService?.synchronisationStatusPu
    .sink { synchronisationStatusSubject.send($0)
    .store(in: &subscriptions)
```

TODO @Vincent Ducastel

SwiftUI

TODO @Vincent Ducastel

DRY (Don't Repeat Yourself)

On définit une variable si plusieurs utilisations OK

Exemple avec : `CoyoteHereGuidance.getInstance().currentState`

DO

```
3  self?.isRunning.accept(currentState == .RUNNING)
4  self?.isNavigation.accept(!(currentState == .STOP
```

DON'T

```
3  self?.isNavigation.accept(!(CoyoteHereGuidance.ge
```


Plus d'informations sur le DRY : DRY

Autres

Abréviations NO RULE

DO

```
1  let coordinates: [String] = []
2  let index: Int
```

```
1  let coord: [String] = []
2  let idx: Int
```

Acronyme NO RULE

DO

```
1  let tta: Int
2  let fcd: Int
```

```
1  let timeToArrival: Int
2  let floatingCarData: Int
```

Extension TODO

Une extension globale

On fait une extension globale pour regrouper des méthodes.

Dans la v12 : On met les extensions dans un module Utils, voir liste des modules de la v12 : Modules

DO

```
1  String+Extension.swift
2  URL+Extension.swift
3  Date+Extension.swift
```

DON'T

```
1  StringExtensions.swift
2  URL+Ext.swift
3  DateExtension.swift
```

Si l'extension commence à grossir, essayer de le découper par profession 👇

Une extension par profession

Coding Guideline v11 Mise à jour : mars 01, 2024 Modifier

DO

```
String+Email.swift

AppCoordinator+DeepLink.swift
```

DON'T

```
String+Extension.swift

AppCoordinatorDeepLinks.swift
```

On les range ou ?

Dans la v12

- On met les extensions dans un module **Utils**, voir liste des modules de la v12 : Modules
- Si l'extension concerne un cas bien précis, on la range et on l'isole dans le module concerné.

Aérer le code

On saute une ligne avant et après le `MARK:` NO RULE

DO

```
internal class GoodService: BaseService {

    // MARK: - Internal properties

    private let property1 = "VALUE_1"
    private let property2 = "VALUE_2"
    private let property3 = "VALUE_3"
}
```

```
class GoodService: BaseService {
    // MARK: - Internal properties
    private let property1 = "VALUE_1"
    privatelet property2 = "VALUE_2"
    privatelet property3 = "VALUE_3"
}
```

Sauter des lignes dans les méthodes NO RULE

DO (preferred)

```
// Example 1
func createCarOptions(_ options: Coyote.RouteOpt
    let routeOptions = createRouteOptions(option
    let avoidanceOptions = createAvoidanceOption

    var carOptions = CarOptions()
    carOptions.routeOptions = routeOptions
    carOptions.avoidanceOptions = avoidanceOptio

    return carOptions
}

// Example 2
```

DO

```
func createCarOptions(_ options: Coyote.RouteOpti
    let routeOptions = createRouteOptions(options
    let avoidanceOptions = createAvoidanceOptions
    var carOptions = CarOptions()
    carOptions.routeOptions = routeOptions
    carOptions.avoidanceOptions = avoidanceOption
    return carOptions
}
```

```
14 func createCarOptions(_ options: Coyote.RouteOpt
15     let routeOptions = createRouteOptions(option
```



```
18
19     carOptions.routeOptions = routeOptions
20     carOptions.avoidanceOptions = avoidanceOptio
21
22     return carOptions
23 }
```

Méthodes simples :

DO (preferred)

```
1  internal func minimizeVolume() -> Bool {
2      return updateVolume(AppVolume.min)
3  }
4
5  internal func minimizeVolume() -> Bool {
6      print("AppVolume.min : \(AppVolume.min)")
7      return updateVolume(AppVolume.min)
8  }
```

DO

```
1  internal func minimizeVolume() -> Bool {
2
3      return updateVolume(AppVolume.min)
4  }
5
6  internal func minimizeVolume() -> Bool {
7
8      print("AppVolume.min : \(AppVolume.min)")
9
10     return updateVolume(AppVolume.min)
11 }
```

Eviter de mettre de la logique dans les paramètres KO

Non validé par l'équipe, pas de règle applicable pour l'instant

> Eviter de mettre de la logique dans les paramètres

Saut de ligne après la première accolade ? NO RULE

DO

```
1  func asTripRoute(online: Bool) -> TripRouteProtoc
2
3      let violatedOptions = getViolatedOptions()
4      let crossingRouteOptions = getCrossingRouteOp
5      //...
```

```
1  func asTripRoute(online: Bool) -> TripRouteProtoc
2      let violatedOptions = getViolatedOptions()
3      let crossingRouteOptions = getCrossingRouteOp
4      //...
```

TODO ?

Compléter les taches :

Description	Date d'échéance	Personne assignée	La tâche apparaît sur
Le `return` avant la completion			Coding Guideline v11

Extension → comment regrouper String, dans un package SPM ? Categorier? créer une extension génère souvent un internal de la classe concerné

https://git.coyotesystems.com/icoyote-ios/app/-/blob/512508ffec497f2f97d7d6b39d4b2fa7ab752d60/iCoyote/mvvm/MapProvider/Routing/RoutingService.swift

- [ ] Le `return` avant la completion

```
return completion([], .noCommunication)

ou

completion([], .noCommunication)
return
```

Idée ?

Pour mettre à jour le document avec un poly dans un channel Teams

Contenu connexe

Coding Guidelines
Coyote iOS
Lue avec ce contenu

Xcode Coding Guidelines (Swift)
Coyote iOS
Lue avec ce contenu

Définition critères de qualité de code
Coyote Android
Plus de contenu similaire

Checklist de qualité de code
Coyote Android
Plus de contenu similaire

Architecture Intégration avec les Libs Alerting v2
Coyote Android
Plus de contenu similaire

Modules
Coyote iOS
Lue avec ce contenu