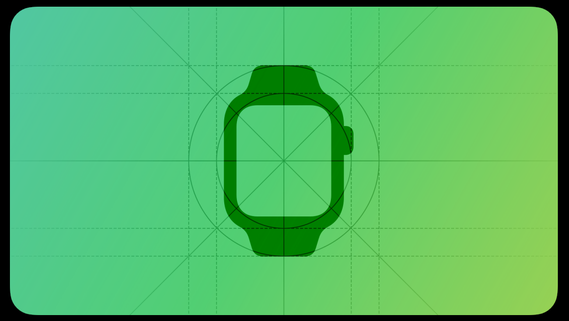


Zoom sur watchOS

Intro à watchOS

Spécificités

- Écran extrêmement petit (38 à 49mm)
- Utilisation "on the go", à quelques dizaines de centimètres de l'écran
- Présence de la Digital Crown, qui permet le scroll sans masquer l'écran
- Durée d'interaction très faible

- Il faut afficher la bonne info au bon moment !

Appareils

- Apple Watch Series 4
- Apple Watch Series 5
- Apple Watch SE
- Apple Watch Series 6
- Apple Watch Series 7
- Apple Watch Series 8
- Apple Watch Ultra

Types d'app

App

Complications

Notifications

Types d'app

- watchOS supporte différents "type d'app"
  - L'application watchOS
  - Les complications
  - Les notifications

Bonnes pratiques

- Afficher les bonnes infos, de manière concise, en un minimum d'interaction
- Limiter la profondeur de navigation, et exploiter la Digital Crown
- Utiliser des complications pour fournir des infos pertinentes
- Utiliser les notifications pour afficher les infos importantes à l'utilisateur

App watchOS

Un peu d'histoire…

- Historiquement dépendante d'un iPhone
  - Code exécuté par l'iPhone, interface affichée par la montre
- Framework UI dédié, WatchKit
  - Simpliste et limité
- Aujourd'hui, les apps tournent sur la Watch et avec SwiftUI !

Un peu d'histoire…

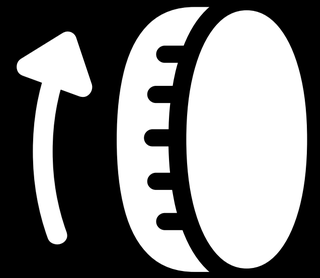
- Une app watchOS peut être autonome, ou liée à une app iOS
- Si liée, une communication peut se mettre en place entre les deux app
  - WatchConnectivity

App watchOS

- Fournir les infos les plus pertinentes en 3 taps max !
- SwiftUI permet une grande flexibilité et réutilisabilité, mais il faut penser "Watch"
  - Pas juste une version miniaturisé de l'app iOS

Digital Crown

- Mode d'interaction très important sur la Watch
  - Permet de sélectionner, faire défiler, ajuster une valeur
- Procure un retour haptique lors de l'utilisation
- Simple à utiliser

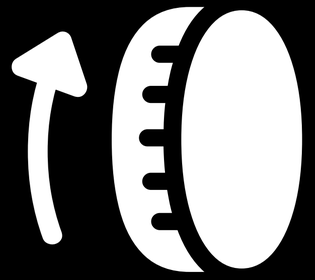


Digital Crown

```
struct SmileView: View {

    @State private var value = 0.0

    var body: some View {
        Text("\(value)")
           .focusable()
            .digitalCrownRotation($value, from: 0.0, through: 100.0, by: 1.0)
    }
}
```

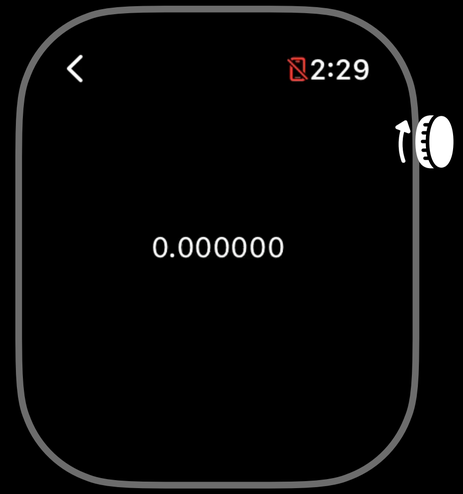


Digital Crown

```
struct SmileView: View {

    @State private var value = 0.0

    var body: some View {
        Text("\(value)")
           .focusable()
            .digitalCrownRotation($value, from: 0.0, through: 100.0, by: 1.0)
    }
}
```



Notifications

Notifications

Notifications

Notifications

Short Look

Long Look

Notifications

Short Look

Custom Long Look

Long Look

Custom Long Look

- WKUserNotificationHostingController

Custom Long Look

```swift
import SwiftUI
import UserNotifications

class NotificationController: WKUserNotificationHostingController<NotificationView> {

    var content: UNNotificationContent?
    var someCustomInfo: String?

    override var body: NotificationView {
        NotificationView(name: someCustomInfo)
    }

    override class var isInteractive: Bool { true }

    override func didReceive(_ notification: UNNotification) {
        content = notification.request.content
        someCustomInfo = content?.userInfo["someCustomInfo"] as? String

        notificationActions = [
            .init(identifier: "directions", title: "Let's go there!")
        ]
    }
}
```

Custom Long Look

```swift
import SwiftUI
import UserNotifications

class NotificationController: WKUserNotificationHostingController<NotificationView> {

    var content: UNNotificationContent?
    var someCustomInfo: String?

    override var body: NotificationView {
        NotificationView(name: someCustomInfo)
    }

    override class var isInteractive: Bool { true }

    override func didReceive(_ notification: UNNotification) {
        content = notification.request.content
        someCustomInfo = content?.userInfo["someCustomInfo"] as? String

        notificationActions = [
            .init(identifier: "directions", title: "Let's go there!")
        ]
    }
}
```

Custom Long Look

```swift
import SwiftUI
import UserNotifications

class NotificationController: WKUserNotificationHostingController<NotificationView> {

    var content: UNNotificationContent?
    var someCustomInfo: String?

    override var body: NotificationView {
        NotificationView(name: someCustomInfo)
    }

    override class var isInteractive: Bool { true }

    override func didReceive(_ notification: UNNotification) {
        content = notification.request.content
        someCustomInfo = content?.userInfo["someCustomInfo"] as? String

        notificationActions = [
            .init(identifier: "directions", title: "Let's go there!")
        ]
    }
}
```

Custom Long Look

```swift
import SwiftUI
import UserNotifications

class NotificationController: WKUserNotificationHostingController<NotificationView> {

    var content: UNNotificationContent?
    var someCustomInfo: String?

    override var body: NotificationView {
        NotificationView(name: someCustomInfo)
    }

    override class var isInteractive: Bool { true }

    override func didReceive(_ notification: UNNotification) {
        content = notification.request.content
        someCustomInfo = content?.userInfo["someCustomInfo"] as? String

        notificationActions = [
            .init(identifier: "directions", title: "Let's go there!")
        ]
    }
}
```

Custom Long Look

```swift
import SwiftUI
import UserNotifications

class NotificationController: WKUserNotificationHostingController<NotificationView> {

    var content: UNNotificationContent?
    var someCustomInfo: String?

    override var body: NotificationView {
        NotificationView(name: someCustomInfo)
    }

    override class var isInteractive: Bool { true }

    override func didReceive(_ notification: UNNotification) {
        content = notification.request.content
        someCustomInfo = content?.userInfo["someCustomInfo"] as? String

        notificationActions = [
            .init(identifier: "directions", title: "Let's go there!")
        ]
    }
}
```

Custom Long Look


```
struct MyWatch_Watch_App: App {

    var body: some Scene {
        WindowGroup {
            ContentView()
       }

        WKNotificationScene(controller: NotificationController.self, category: "categoryA")
        WKNotificationScene(controller: OtherNotificationController.self, category: "categoryB")
    }
}
```

Complications

A complication displays timely, relevant information on the watch face, where people can view it each time they raise their wrist.

Apple Human Interface Guidelines - Complications

Complications

Complications

- Historiquement construites avec ClockKit
- WidgetKit prend le relais à partir de watchOS 9
- WidgetKit a été construit en s'inspirant de ClockKit, donc beaucoup de similitudes