нетология

Light/Dark тема

Алексей
Жембловский

Алексей Жембловский

iOS developer EPAM

План занятия

1. Light/Dark тема
2. Про цвета и картинки
3. Адаптируем view компоненты под темы
4. Кастомные темы
5. UX-приложения при работе с темами

Light/Dark

Что дает нам поддержка тёмной темы:

- Забота о зрении пользователя (в ночное время)
- Экономия заряда батареи (на LED дисплеях)
- Альтернативный дизайн (возможность выбора для пользователя)

Обратная сторона поддержки тёмной темы:

- Труднее делать дизайн для приложения
- Адаптация занимает много времени для приложения с одной темой
- Необходимость в новой палитре

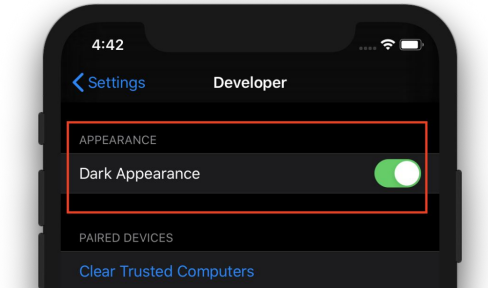
Как работать с темами

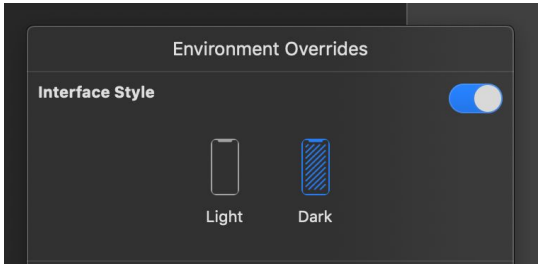
По умолчанию любой новый проект поддерживает обе темы, чтобы явно работать только с одной темой можно:

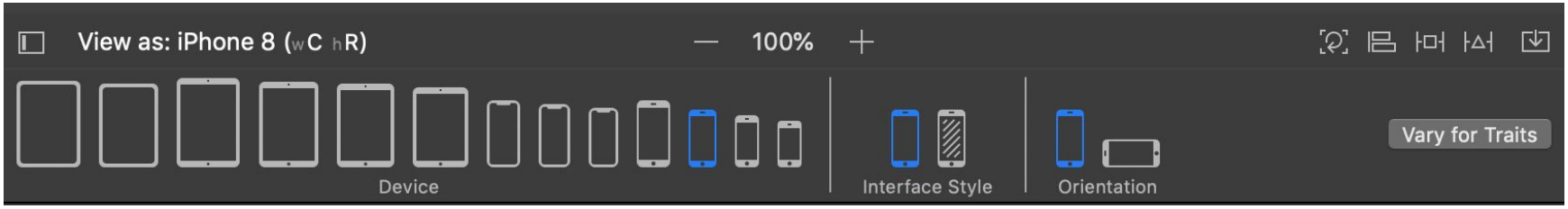
- **Глобально: выбрать тему приложения в Info.plist,** выставляем у ключа `UIUserInterfaceStyle` значение `Light` или `Dark`
- **Локально для View, ViewController или window:** выставляем свойство `overrideUserInterfaceStyle = .dark` или `.light`

Отладка поддержки разных тем

- Включаем тёмную тему на симуляторе
- Смотрим темную тему в сториборде
- Меняем окружение в preview (только для Swift UI)







Новые семантические цвета

Цвета, используемые по назначению:

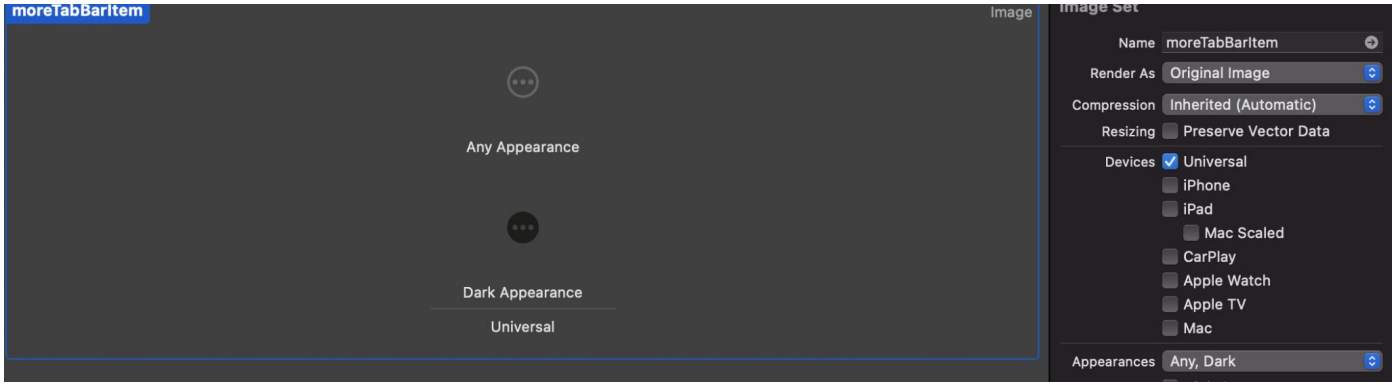
Color	Used for	API
Label	A text label that contains primary content.	label
Secondary label	A text label that contains secondary content.	secondaryLabel
Tertiary label	A text label that contains tertiary content.	tertiaryLabel
Quaternary label	A text label that contains quaternary content.	quaternaryLabel
Placeholder text	Placeholder text in controls or text views.	placeholderText
Separator	A separator that allows some underlying content to be visible.	separator
Opaque separator	A separator that doesn't allow any underlying content to be visible.	opaqueSeparator
Link	Text that functions as a link.	link

Добавляем свои цвета, изменяем картинки

Добавляем Color asset (iOS 11+), указываем нужные цвета

Опционально: указываем цвета для высокой контрастности

Меняем изображения для каждой темы



Добавляем свои цвета

Создаем свои цвета через код:

```
static var tint: UIColor = {
    if #available(iOS 13, *) {
        return UIColor { (traitCollection: UITraitCollection) -> UIColor in
            if traitCollection.userInterfaceStyle == .dark {
                return UIColor.someDarkColor // Темный цвет из палитры
            } else {
                return UIColor.someLightColor // Светлый цвет из палитры
            }
        }
    } else {
        return UIColor.someDefaultColor // Цвет по умолчанию
    }
}()
```

Добавляем свои цвета

Удобный extension для создания адаптивных цветов:

```
extension UIColor {
    static func createColor(lightMode: UIColor, darkMode: UIColor) -> UIColor {
        guard #available(iOS 13.0, *) else {
            return lightMode
        }
        return UIColor { (traitCollection) -> UIColor in
            return traitCollection.userInterfaceStyle == .light ? lightMode :
darkMode
        }
    }
}
```

Адаптируем view компоненты под темы

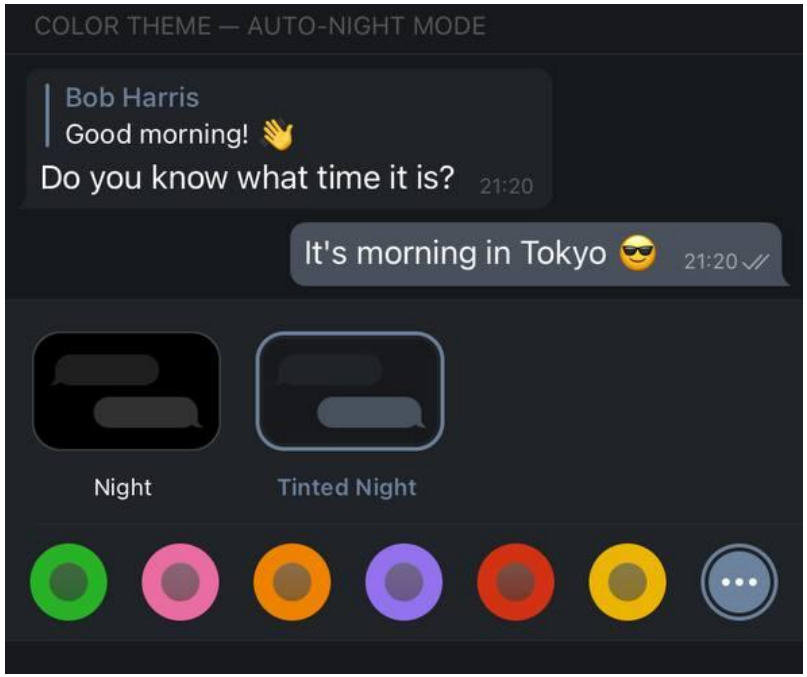
- При изменении темы система перерисовывает каждое окно и view. Происходит изменение traitEnvironment, после этого вызываются следующие методы.
- Для того, чтобы интерфейс правильно реагировал на изменение темы, нужно реализовать методы с учетом темы. По большей части все изменения происходят автоматически, но, например, при работе с CALayer нужно вручную применять изменения:

UIView	traitCollectionDidChange(_:) layoutSubviews() draw(_:) updateConstraints() tintColorDidChange()
UIViewController	traitCollectionDidChange(_:) updateViewConstraints() viewWillLayoutSubviews() viewDidLayoutSubviews()

Light/Dark тема в iOS - demo

Кастомные темы

Важно уметь порадовать пользователей старых операционных систем, а также дать возможность использовать другие цветовые схемы. Как это можно сделать?



Кастомные темы

Пошагово:

- Создаем набор тем и цветовые палитры для каждой
- Подписываем наши ui-компоненты на изменение темы (через нотификации или таблицу слушателей)
- При изменении темы перерисовываем компоненты

Триггером к изменению темы на iOS 12 и ниже будет изменение в настройках приложения.

Для iOS 13 и выше – тоже изменение в настройках приложения + изменение в настройках системы.

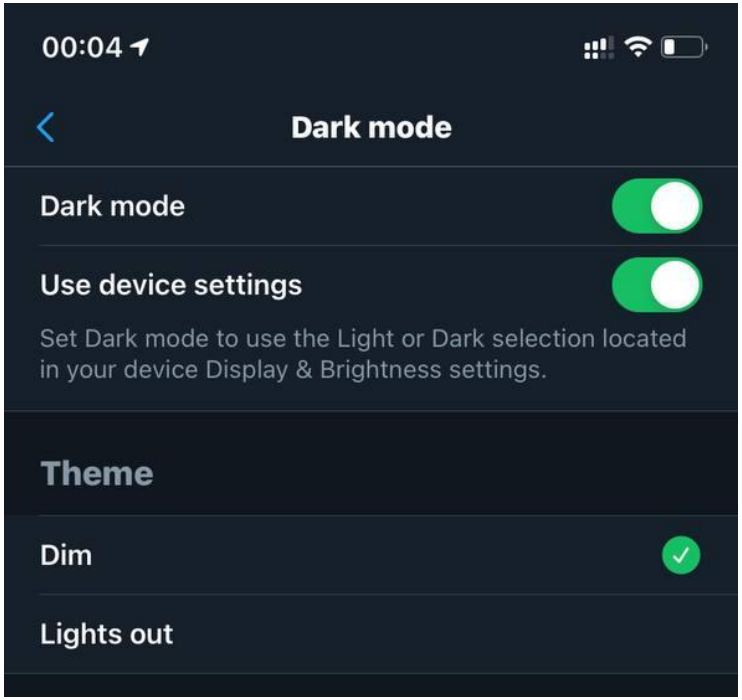
Кастомные темы

```
// Объявляем структуру темы (тип темы и цветовая палитра)
struct Theme {
    enum ThemeType {
        case light
        case dark
        @available(iOS 13.0, *)
        case adaptive
    }
    let type: ThemeType
    let colors: ColorPalette
}
// Объявляем набор доступных тем
extension Theme {
    static let light = Theme(type: .light, colors: .light)
    static let dark = Theme(type: .dark, colors: .dark)
    @available(iOS 13.0, *)
    static let adaptive = Theme(type: .adaptive, colors: .adaptive)
}
// Протокол для адаптации всем view компонентам
protocol Themeable: class {
    func apply(theme: Theme)
}
```

UX приложения при работе с темами

Лучшее решение при работе с темной темой – дать пользователю возможность настраивать ее самому или позволить системе сделать это за него.

Также здорово дать возможность выбирать темы (например, более контрастную).



Итоги

- В современных приложениях поддержка темной темы является опциональным, но все-таки желательным элементом;
- Поддержав ее, можно увеличить аудиторию приложения и сделать его более привлекательным для большего числа пользователей;
- Внедрение темы может затянуться на продолжительное время в зависимости от размера приложения. Лучше учитывать это с самого начала разработки проекта или начинать внедрять тему как можно раньше.

Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задавайте **в чате** учебной группы.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Алексей Жембловский