Вступ

У наші дні в інтернеті можна знайти безліч різноманітних мобільних додатків. Кожен з них був створений для деякої мети:

* Гра
* Клієнт деякого сервісу
* Корисна программа
* Інші варіанти

Зараз однією з найперспективніших напрямів ринку IT - є розробка мобільних додатків. Тому в цій роботі я покажу як створити мобільний клієнт для інтернет-сервісу, описуючи власний досвід створення такого мобільного додатку.

Вибір засобів розробки

Зараз є два основні варіанти розробки мобільних додатків:

* Писати нативний код для кожної з платформ
* Використовувати кросплатформений фреймворк для розробки

Другий варіант є кращим, бо не потрібно переписувати весь код для переносу додатку на іншу платформу.

З мінусів такого підходу можна виділити низьку швидкість роботи додатку, але цей мінус вже майже усунено в сучасних фреймворках.

Моїм вибором став фреймворк React Native, бо я завжди вважав React - перспективною розробкою, та не було часу розібратися в ній.

React Native

Загальна інформація

З офіційного сайту: “React Native - інструмент, що дозволяє створювати мобільні додатки використовуючи лише Javascript. Використовує однаковий дезайн з React, маючи декларативний стиль опису користувацького інтерфейсу з компонентів”



Додаток з React Native - це реальний мобыльний додаток

Використовуючи React Native, ви створюєте не “мобільний веб додаток”, “HTML5 додаток” або “гібридний додаток”. Ви створюєте справжній додаток, який не відрізнити від додатку, написаного на Java або Objective-C.

React Native використовує ті самі фундаментальні компоненти користувацького інтерфейсу, що й звичайні iOS або Android додатки. Ви просто кладете до купи ці нативні компоненти використовуючи JavaScript та React.

Реальні приклади

Ось приклади мобільних додатків, побудованих на React Native:

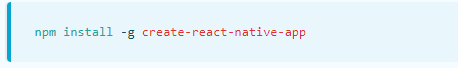
* Facebook
* Instagram
* MakeItOpen
* Bloomberg
* Pinterest
* Skype
* Tesla
* Uber
* Walmart

Налагодження середовища розробки

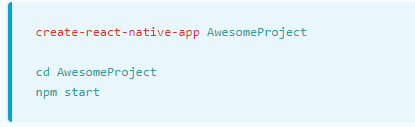
Для написання коду можна використовувати будь-який текстовий редактор. Я використовував Sublime Text.

Для розробки на React Native треба встановити Nodejs та npm. Версія JavaScript - ES2015 ( також відома як ES6).

Тепер використовуємо наступні команди:



Щоб створити проект з назвою “AwesomeProject” достатньо написати:



Останній рядок запустить сервер для розробки.

React Native дозволяє зручно тестувати додаток на мобільному пристрої. Для цього його не треба постійно перевстановлювати. Після запуску останньої команди ми побачимо наступну картину:



Тепер на мобільному пристрої встановлюємо додаток Expo, запускаємо його, там є можливість сканувати QR код. Скануємо код з консолі і через декілька секунд бачимо працюючий прототип додатку на мобільному пристрої.

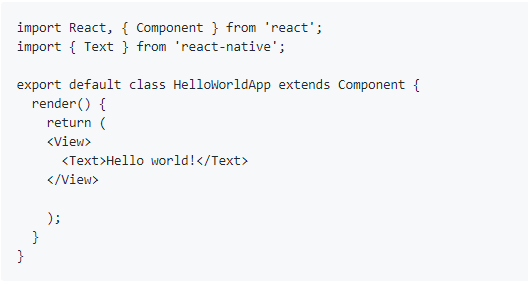
При зміні коду та збереженні змін (натискання Ctrl+S у редакторі) до Expo буде надіслано остання версія додатку і через декілька секунд можна тестувати оновлену версію. Дуже зручно.

Базова структура React

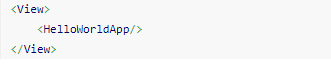
Компоненти

React Native дуже схожий на React, просто використовує інші компоненти для побудови інтерфейсу.

Взагалі увесь React - це інструмент для побудови реактивних користувацьких інтерфейсів. Усе будується через компоненти, які є своєрідними будівельними блоками. Ось приклад коду компоненту, який виводить текст “Hello world!”:

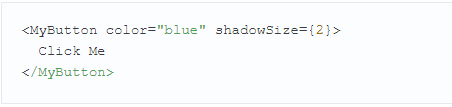


Тут <View> - це теж деякий компонент. Щоб потім десь використати написаний компонент HelloWorldApp треба лише написати:

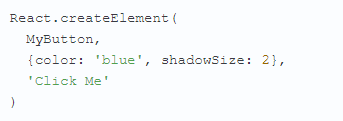


У коді компоненту можна було помітити метод render(). Він викликається, коли потрібно перемалювати компонент. У методі render() бачимо трохи незвичайний return. У нього всередині так званий JSX. Формально, JSX - це просто синтаксичний цукор для функції React.createElement(component, props, …children).

Наступний код:



Перетворюється в:



Крім render() у будь-якого компоненту є ще інші методи та конструктор.

Також у будь-якого React компоненту є дві базові складові:

* props
* state

Props

Props - це параметри які буди передані в компонент.



Тепер у коді компоненту можна використовувати параметр this.props.name (у цьому прикладі, параметр буде мати значення ‘John’)

Щоб не помилитися з заданням цих параметрів та їх типів існує чудова бібліотека PropTypes. Після опису компоненту можна додати наступний блок:



Так для чого ж потрібні ці props? Взагалі props - це зовнішні параметри компонента, при їх зміні компонент буде перемальовано ( про це чуть пізніше). Якщо б ми написали у компоненті HelloWorldApp такий код:



То при зміні параметру name, змінювався б і відображуваний текст. ( У нашому випадку було б “Hello, John!”).

Також в props можна передавати фунції. Тоді компонент при виникненні деяких ситуацій викликувати ці функції. (Наприклад колбек-функція натиску на кнопку).

State

State - внутрішній стан компоненту. По суті, this.state - це об’єкт, який можна використовувати для збереження якихось даних, які впливають на компонент. Наприклад, у компонента checkBox у стані можна зберігати статус чекбокса (checked або unchecked). При зміні state компонент буде перемальовано.

Є один нюанс. Напряму змінювати this.state неможна (можна, але компонент не перемалюється). Для цього є метод this.setState(newState), який замінює поточний state на newState (поля які не встановлені в newState будуть взяті зі старого state).

Наприклад приклад простого компоненту - личильника:



Підсумок

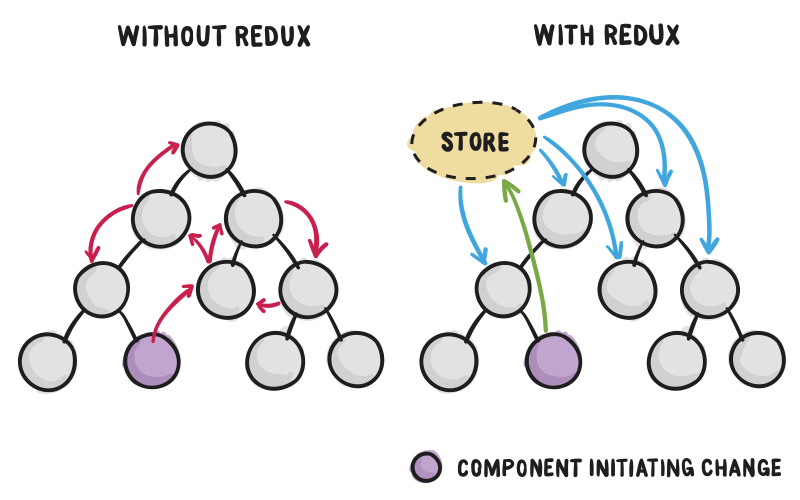
Використовуючи props і state можна написати компонент будь-якої складності, просто розбивши його на дрібніші компоненти. Це основа розробки на React.

Redux

Як можна було помітити, до цього часу ми навчилися лише будувати компоненти, які не несли загальної бізнес-логіки. Для цього потрібен якийсь “загальний стан програми”, щоб його можна було змінювати відповідно до ідей бізнес-логіки та реагувати на ці зміни. Накращою біблотекою, яка дає такий “глобальний стан” є Redux.

А для чого взагалі цей “глобальний стан” потрібен?

Порівняйте:



Тобто бачимо, що без Redux зміна деякого компоненту викликає хаотичну зміну в інших, а з Redux - спочатку змінюється store, а потім усі залежні від цієї зміни компонентіи.

Щоб використовувати Redux треба зрозуміти, як проходить взаємодія з глобальним store.

Є дві базові складові розробки на Redux:

* Actions
* Reducers

Actions

Actions - опис дії, яка змінює store. В загальному випадку це просто об’єкт з параметром type та іншими додатковоми параметрами.

Щоб виконати деякий action є метод dispatch(action).

Reducers - такі блоки коду, які на основі отриманого action змінюють store.

Підсумок

Redux - потужна бібліотека для підвищення швидкості розробки та масштабованості програми. Data flow стає значно зрозумілішим.

Детальніше можна почитати на офіційному сайті.

Планування архітектури проекту

Темою мобільного додатку було вибрано клієнт для соціальної мережі. Основні екрани, які користувач зможе побачити було вибрано наступними:

* Екран завантаження
* Екран логіну
* Екран регістрації
* Екран списку людей в мережі, з якими у користувача є хоча б одна мова
* Екран списку людей в мережі, які знаходяться у зазначеному радіусі від корисувача
* Екран профайлу користувача
* Екран налаштувань додатку
* Екран діалогів
* Екран конкретного діалогу