WEBINAR

React Native Jumpstart

mark@objectcomputing.com

WHY?

- Want to develop native apps for Android and iOS
- Don't have high performance requirements (not a video game)
- Want most of the code to work on both
- Don't want to learn Java, Kotlin, or Swift
- Want watch and live reload (provided by Expo)
- Maybe already know React

SAME IN REACT AND REACT NATIVE

- Language - JS or TS
- Components - how defined
- Props
- State management
- Lifecycle methods
- Sending HTTP requests
- Linting tools - ESLint
- Formatting tools - Prettier
- Unit testing - Jest?

DIFFERENT IN REACT AND REACT NATIVE

- Elements used - div vs. View, ...
- Platform-specific components - Android and iOS
- Styling approach - CSS vs. style objects
- Debugging approach - web browser vs. simulators & Expo
- Deployment - web server vs. app stores
- Access to device capabilities - ex. camera
- End-to-end testing - Cypress vs. ?
- Devtools - browser extensions vs. simulators

RESOURCES

- React Native - https://facebook.github.io/react-native/
- Awesome React Native - http://www.awesome-react-native.com/
- ADD MORE!

EXPO (v32)

- Free tool that wraps React Native and adds many features
- https://expo.io/
- Runs on macOS, Windows, and Linux
- Supports Android 5+ and iOS 10+
  - don't use if older versions must be supported
- Each version of Expo works with a specific version of React Native
- Simplifies updating to new versions of React Native

EXPO MODES

OBJECT COMPUTING

- "managed"
  - no need to install Android Studio or Xcode
  - after code changes, Expo rebuilds app, hosts in local server, and deploys to simulators/devices
  - some native APIs are not supported
  - **`expo upload`** deploys to stores
  - **`expo publish`** deploys an update to stores
- "bare"
  - can use all native APIs and include native code (Java, Kotlin, Swift)
  - you handle steps to build, upload, and publish using Android Studio and Xcode

examples include Bluetooth, in-app purchases, and WebRTC

devices must have "Expo Client" installed from app store

for details on how it works, see https://docs.expo.io/versions/v32.0.0/workflow/how-expo-works/

EXPO MANAGED MODE (does not apply to "bare" mode)

OBJECT COMPUTING

- Pros
  - live reload in development
  - testing on real devices using Expo Client
  - creates binaries without interacting with Android Studio or Xcode
    - in "managed" mode, not "bare" mode
  - supports over-the-air updates to apps in stores
- Cons
  - no support for native modules
    - must "eject" to use native code
  - no background code execution
  - results in larger apps
    - ~15MB for Android
    - ~20MB for iOS

but these sizes are close to the average size of mobile apps

EXPO COMPONENTS AND APIS

- Components
  - **`Camera`**, **`MapView`**, **`Svg`**, **`Video`**, and more
- APIs
  - AppAuth, Audio, Calendar, Contacts, DeviceMotion, FaceDetector, Font, Haptics, ImagePicker, Localization, Location (geolocation), MailComposer, MediaLibrary, Notifications, Payments, Permissions, Print, SecureStore, Sensors, SMS, Speech, SQLite, Updates, and more

see "API Reference" in left nav. at https://docs.expo.io/versions/latest/

Which of these support push notifications?

SETUP

- There are many steps required to get started

- Step 1: install Node.js from https://nodejs.org/

OPTION #1 - React Native CLI ... skip ahead to option #2

- Browse https://facebook.github.io/react-native/docs/getting-started
- Click "React Native CLI Quickstart"
- Select development OS and target OS
- Follow instructions

- **`react-native init MyProject`**
- **`cd MyProject`**

... OPTION #1 - React Native CLI

OBJECT COMPUTING

- To run on iOS simulator
  - **`react-native run-ios`**
  - takes about 5 minutes the first time!
  - no live reload; press cmd-R in simulator
- To run on Android simulator
  - start simulator as shown on slide 13
  - create **`local.properties`** file
    - in ***`project-directory`*`/android`**
    - add line **`sdk.dir = /Users/USERNAME/Library/Android/sdk`**
  - **`react-native run-android`**
  - no live reload; press R twice in simulator

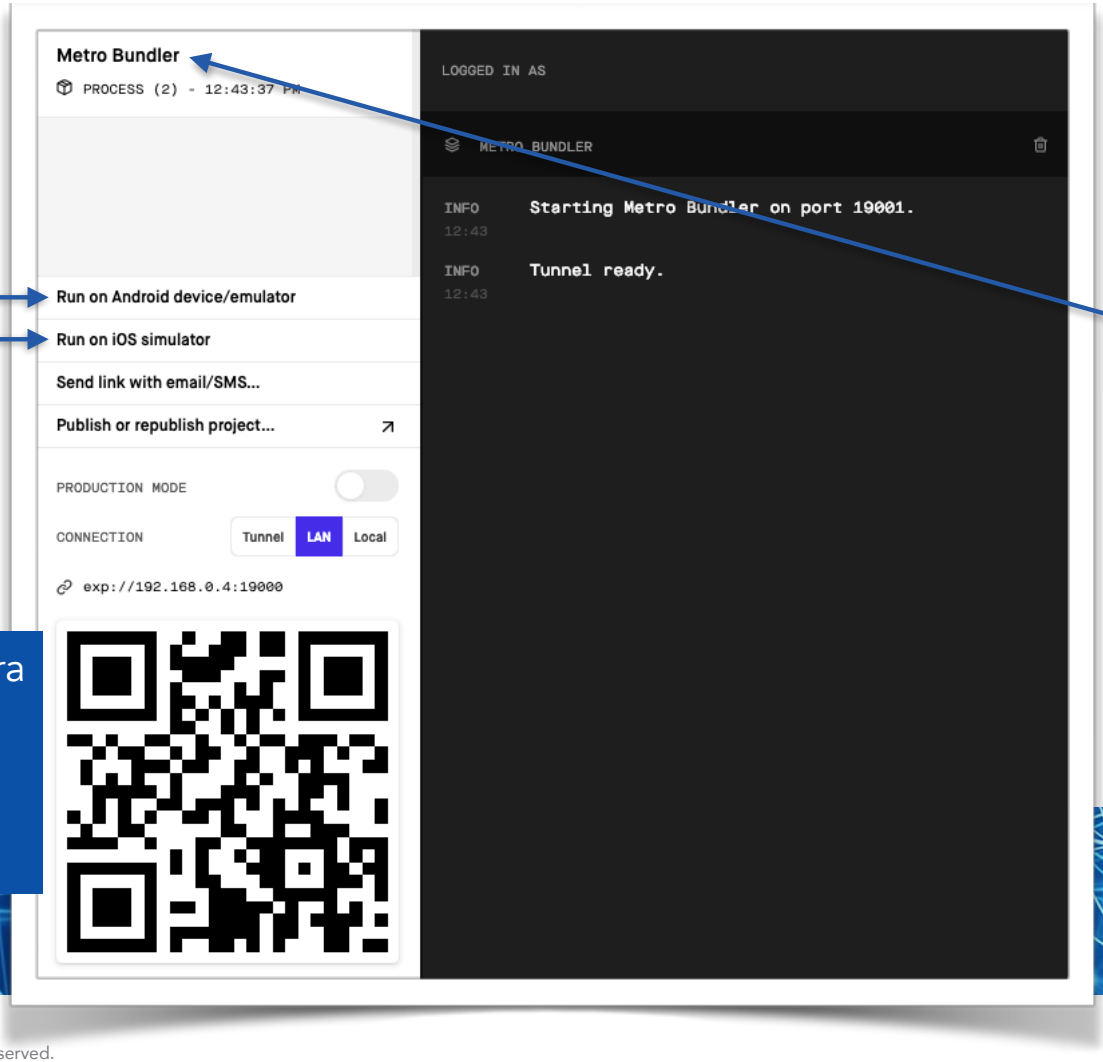
OPTION #2 - Expo CLI our focus

OBJECT COMPUTING

- **`npm install -g expo-cli`**
- **`expo init my-project`**
  - choose a template, “blank”, “tabs”, or “bare-minimum”
  - enter app name as it should display on devices
  - enter “Y” to install dependencies including react-native and much more
- **`cd my-project`**
- **`npm start`** or **`yarn start`** or **`expo start`**
  - opens a browser tab shown on next slide

expo v32.0.0 requires restart of this and simulators if a JS syntax error is saved!

Expo App Page



click these to run on a simulator that is already running

scan with a device camera to run app on device in "Expo Client"
(may need to switch connection to "Tunnel")

"Metro Bundler" is a JavaScript bundler for React Native that supports watch and live reload in Android and iOS simulators.

OBJECT COMPUTING

SIMULATORS

- Can use Android and iOS simulators to test apps
- iOS simulator requires macOS
  - true? See “Xcode for Windows”

INSTALL ANDROID SIMULATOR

- Browse https://developer.android.com/studio
- Press “DOWNLOAD ANDROID STUDIO”
- Double-click downloaded installer and follow instructions

RUN ANDROID SIMULATOR

- Launch Android Studio app and follow one-time setup instructions
- Select Configure ... AVD Manager stands for "Android Virtual Device"
- Click "+ Create Virtual Device..." to add a new device simulator
- Click "Download" after an existing device simulator
- Click green triangle "play" button after a device simulator
  - takes a couple of minutes the first time
- Can quit Android Studio

INSTALL AND RUN IOS SIMULATOR

OBJECT COMPUTING

- Install Xcode
  - browse https://developer.apple.com/xcode/ and click Download button
- To run simulator
  - launch Xcode app
  - select Xcode ... Open Developer Tool ... Simulator
  - select device by selecting Hardware ... Device ... *os* ... *device-name*
  - can quit Xcode

Alternative to launching Xcode:

```
npm install -g ios-sim
ios-sim showdevicetypes
ios-sim start --devicetypeid "full-device-type"
```

Handy npm script:

```
"ios-sim": "ios-sim start --devicetypeid 'iPhone-XR, 12.1'",
```

RUN APP ON SIMULATOR

OBJECT COMPUTING

- Start one or both simulators
- In Expo web app, press
  "Run on Android device/emulator" or "Run on iOS simulator"
  - takes a couple of minutes the first time
  - installs Expo Client in simulator
  - in iOS simulator, press cmd-d for developer menu

INSTALL EXPO CLIENT ON DEVICES

- Install “Expo Client” app on mobile devices
- Launch “Expo Client”
- Press “Sign up for Expo”
- Enter requested info. and press “Sign Up”
  - will receive an authentication email

RUN APP ON ANDROID DEVICE USING EXPO CLIENT

- TRY THIS AND DOCUMENT!

RUN APP ON iOS DEVICE USING EXPO CLIENT

- Open Camera app on device
- Point at QR code in Expo web app
- Press "Open in Expo"
  - will launch Expo Client app if not already running

When "CONNECTION" is "LAN", computer and phone must be on same Wifi network.
May work better when "CONNECTION" is "Tunnel".

WATCH AND LIVE RELOAD

- Changes to files in app trigger live reload in all simulators and in Expo Client on devices

INITIAL FILES TO EDIT

- **`package.json`** - dependencies and scripts
- **`app.json`** - app configuration
- **`app.js`** - top-most component

REACT NATIVE DOES NOT USE HTML

- `div` -> `View`
- `span` -> `Text`
- `button` -> `Button`
- `img` -> `Image`

DEBUGGING

- `console.log` output
  - goes to terminal where development server is running
  - when using Expo, this is in browser tab
- In simulator
  - cmd-d; Toggle Element Inspector; click an element
  - can't enter text in Input elements when this is on

PROVIDED CROSS-PLATFORM COMPONENTS ...

- Containers: **`View`**, **`ScrollView`**, **`FlatList`**, **`SectionList`**
- Output: **`Text`**, **`Image`**
- Input: **`Button`**, **`TextInput`**, **`Switch`**, **`Picker`**, **`Slider`**
- Styling: **`StyleSheet`**

`View` and **`ScrollView`** automatically use flexbox for layout where **`flexDirection`** defaults to "**`column`**".

... PROVIDED CROSS-PLATFORM COMPONENTS

- **`ActivityIndicator`**
  - loading indicator
- **`Alert`** dialog
- **`KeyboardAvoidingView`**
  - moves out of way of virtual keyboard
- **`Modal`** dialog
- **`RefreshControl`**
  - provides "pull to refresh" inside **`ListView`** or **`ScrollView`**
- **`StatusBar`**
  - controls app status bar
- **`WebView`**

PROVIDED LIBRARIES

- **`Animated`**
  - create animations
- **`CameraRoll`**
  - save to photo library and get specified photos
- **`Clipboard`**
  - get and set clipboard string
- **`Dimensions`**
  - get/set screen dimensions and listen for changes
- **`PixelRatio`**
  - get pixel density and font scaling factor

iOS-SPECIFIC COMPONENTS

- ActionSheetIOS
- AlertIOS
- DatePickerIOS
- ImagePickerIOS
- NavigatorIOS
- ProgressViewIOS
- PushNotificationIOS
- SegmentedControlIOS
- TabBarIOS

ANDROID-SPECIFIC COMPONENTS

- BackHandler
- DatePickerAndroid
- DrawerLayoutAndroid
- PermissionsAndroid
- ProgressBarAndroid
- TimePickerAndroid
- ToastAndroid
- ToolbarAndroid
- ViewPagerAndroid

NAVIGATION

- Like routing in a React app
- Options to manage navigation between screens include
  - **React Navigation**
    - cross-platform library at https://facebook.github.io/react-native/docs/navigation#react-navigation
    - easy to use
  - **React Native Navigation**
    - cross-platform library at https://github.com/wix/react-native-navigation
    - not as easy, but provides native look and feel

WEB VIEW

- Cross-platform rendering of HTML
- https://github.com/react-native-community/react-native-webview
- Replacement for provided **`WebView`** which will be removed
- Not supported by Expo unless ejected

THIRD-PARTY COMPONENTS

- NativeBase at https://nativebase.io
- Current list of components
  - Anatomy, Accordion, ActionSheet, Badge, Button, Card, Check Box, Date Picker, Deck Swiper, FABs, Footer Tabs, Form, Header, Icon, Layout, List, Picker, Radio Button, Search Bar, Segment, Spinner, Swipeable List, Tabs, Thumbnail, Toast, Typography, Drawer, Ref

STYLING

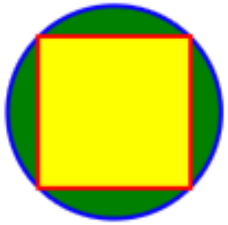
OBJECT COMPUTING

- Supports a subset of CSS
- Units (ex. px) are not specified and depend on element and property
- **`Stylesheet.create`** validates CSS properties and values
- Most layout is done with flexbox
  - one difference: triggered with **`flex`** property, not **`display: flex`**

SVG ...

- Can use react-native-svg on npm
  - https://github.com/react-native-community/react-native-svg#common-props
- Installing
  - when using Expo, skip this because it is already installed
  - **`npm install react-native-svg`**
  - **`react-native link react-native-svg`**
- Example
  - see next slide

... SVG



```
import {Svg} from 'expo';
import React from 'react';
import {StyleSheet, Text, View} from 'react-native';

const {Circle, Rect} = Svg;

const SIZE = 100;
const HALF_SIZE = SIZE / 2;
const OFFSET = SIZE * 0.15;
const SIZE7 = SIZE * 0.7;

const SvgDemo = () => (
  <View style={styles.container}>
    <Svg
      height="100%"
      width="100%"
      viewBox={`0 0 ${SIZE} ${SIZE}`}>
      <Circle
        cx={HALF_SIZE}
        cy={HALF_SIZE}
        r={HALF_SIZE - 1}
        stroke="blue"
        strokeWidth={2}
        fill="green"
      />
      <Rect
        x={OFFSET}
        y={OFFSET}
        width={SIZE7}
        height={SIZE7}
        stroke="red"
        strokeWidth={2}
        fill="yellow"
      />
    </Svg>
  </View>
);

const styles = StyleSheet.create({
  container: {
    alignItems: 'center',
    justifyContent: 'center',
    height: SIZE,
    width: SIZE
  }
});

export default SvgDemo;
```

CAMERA

- Doesn't work in simulators, but does in Expo Client on mobile devices

DEPLOYING TO APP STORES

- Many steps are required
- Summarized at https://apiko.com/blog/deploying-react-native-apps-to-app-store-and-play-market/
- Developer accounts are not free
  - iOS - need an Apple Developer account which is $99 per year
  - Android - need a ? which is a $25 one-time registration fee

WRAP UP

- Add this!

LEARN MORE ABOUT OCI EVENTS AND TRAINING

Events:

- objectcomputing.com/events

Training:

- objectcomputing.com/training
- grailstraining.com
- micronauttraining.com

Or email info@ocitraining.com to schedule a custom training program for your team online, on site, or in our state-of-the-art, Midwest training lab.

OBJECT COMPUTING

CONNECT WITH US

1+ (314) 579-0066

@objectcomputing

objectcomputing.com