OBJECT
COMPUTING

WEBINAR

React Native Development

mark@objectcomputing.com

SOURCE FILES

- Expo expects to find App.js in top project directory
  - see import in **`node_modules/expo/AppEntry.js`**
- Can place other source files in **`src`** subdirectory

DEBUGGING ...

OBJECT COMPUTING

- To open debug page
  - on device, shake
  - in Android simulator press ctrl-m (cmd-m)
  - in iOS simulator press ctrl-d (cmd-d)
- `console.log` output
  - goes to terminal where development server is running
  - to see in Chrome devtools Console, open debug page, click “Debug Remote JS” to get new browser tab, and press ctrl-shift-i (cmd-option-i)

... DEBUGGING

- To inspect element hierarchy, size, and CSS properties
  - open debug page, tap “Toggle Element Inspector”, and click an element
  - can’t change input element values when this is open

PERFORMANCE

- To evaluate performance
  - open debug page and tap “Show Perf Monitor”
  - displays a white rectangle in upper-left that shows
    - process memory usage (RAM)
    - JavaScript thread memory usage (JSC)
    - # of **`View`** elements currently visible and saved in memory
    - main thread frame rate (UI)
    - JavaScript thread frame rate (JS)

WATCH AND LIVE RELOAD

OBJECT COMPUTING

- Changes to files in app can trigger reload in all simulators and in Expo Client on devices if enabled
- Live Reload vs. Hot Reloading
  - can enable only one
  - live reload loses state
  - hot reloading retains state
- To enable, open developer menu and tap either "Enable Live Reload" or "Enable Hot Reloading"

REACT-DEVTOOLS

OBJECT COMPUTING

- To install
  - `npm install -g react-devtools`
- To start
  - run app in a simulator
  - enter **`react-devtools`**
  - opens an Electron app

- To use
  - enter name of a React component in search box
  - expand to see child components
  - click a component to see its props, state, and style
  - for more see https://www.npmjs.com/package/react-devtools

can change CSS and some props and state

When the Element Inspector is open, clicking an element in the simulator causes react-devtools to highlight the element and show its props, state, and style.

REFRESHER ON REACT PROPS AND STATE

OBJECT COMPUTING

props	state
passed from parent	created in component
immutable	mutable by component
parent can pass different values	component methods can change, but these can be passed to children

PLATFORM DIFFERENCES

OBJECT COMPUTING

- Guidelines differ between Android and iOS
- Consider these to provide the most native experience
- Resources
  - https://medium.muz.li/differences-between-designing-native-ios-apps-and-native-android-apps-e71256dfa1ca
  - https://www.ready4s.com/blog/android-vs-ios-comparing-ui-design
  - https://medium.com/@vedantha/interaction-design-patterns-ios-vs-android-111055f8a9b7
  - https://medium.com/@chunchuanlin/android-vs-ios-compare-20-ui-components-patterns-part-1-ad33c2418b45

PLATFORM-SPECIFIC CODE

OBJECT COMPUTING

- Use **`Platform`** module to take platform-specific actions
- **`Platform.OS`**
  - holds string **`'android'`** or **`'ios'`**
- **`Platform.select(obj)`**
  - ***`obj`*** has properties **`android`** and **`ios`**
  - returns value of property that matches current platform
  - often used in CSS property lists
    - values are objects that are used with spread operator

```
const company =
  Platform.OS === 'android' ?
  'Google' : 'Apple';
```

```
const styles = StyleSheet.create({
  container: {
    color: 'white',
    ...Platform.select({
      android: {
        backgroundColor: ''
      },
      ios: {
        backgroundColor: ''
      }
    })
  }
});
```

PLATFORM-SPECIFIC CODE ...

OBJECT COMPUTING

- One way to implement platform-specific code is to use platform file extensions
  - **`___.android.js`** and **`____.ios.js`**
  - in other files, import just file name with no extension
  - for example, the **`.android.js`** file could use **`DatePickerAndroid`** and the **`.ios.js`** file could use **`DatePickerIOS`**

... PLATFORM-SPECIFIC CODE

OBJECT COMPUTING

- Another way to implement platform-specific code is to use **`Platform.OS`** or **`Platform.select`**
  - **`Platform.OS`** will be the string **`'android'`** or **`'ios'`**
  - can use **`Platform.select`** with spread operator in style objects

```
Platform.select({
  android() {
    // Android-specific code goes here.
  },
  ios() {
    // iOS-specific code goes here.
  }
});
```

EXPO SNACK ...

- For experimenting with React Native in a browser
- No need to install anything
- Like Codepen and JSFiddle for React Native
- https://blog.expo.io/sketch-a-playground-for-react-native-16b2401f44a2

... EXPO SNACK

OBJECT COMPUTING

- To use
  - browse https://snack.expo.io/ and log in
  - run “Expo Client” app on mobile device
  - click “Projects” and note Device ID
  - enter Device ID in web UI
  - click “QR Code” tab
  - scan with device camera
  - app will appear on mobile device
  - modify app code in browser; no need to save
  - will hot reload on device
- To export
  - press Save button
  - click Export
  - will download zip file containing code

HTTP SUPPORT

OBJECT COMPUTING

- React Native supports the Fetch API
- Doesn't allow http requests; must use https
  - unless configured to allow use of http

SWAPI

- Star Wars REST API good for building demo apps
- https://swapi.co