OBJECT
COMPUTING

WEBINAR

React Native Styling

mark@objectcomputing.com

STYLING ...

OBJECT COMPUTING

- Supports a subset of CSS
  - properties supported vary by component
  - some are specific to Android or iOS
- Units (ex. px) are not specified
  - and depend on element and property
- **`Stylesheet.create`**
  - validates CSS properties and values

```
const styles = StyleSheet.create({
  container: {
    backgroundColor: 'cornflowerblue',
    flex: 1, // fills screen vertically
    justifyContent: 'flex-start',
    alignItems: 'center'
 },
  error: {
    color: 'red',
    fontSize: 36,
    fontWeight: 'bold'
  },
});
```

definition of **`styles`** object typically appears after component definition

typical to use **`styles.container`** for top element in each component

```
<ScrollView contentContainerStyle={styles.container}>

<Text style={styles.error}>
  Authentication failed!
</Text>
```

... STYLING

OBJECT COMPUTING

- Inline styling
  - `style={{prop1: value1, prop2: value2, ...}}`
- Combining multiple style objects
  - `style={[styleObject1, styleObject2, ...]}`
  - when multiple style objects define same property, last one in wins
  - can also combine outside **`style`** prop

```
const newStyles = StyleSheet.flatten([styleObject1, styleObject2, ...]);
```

GOTCHA!
The prop name is "**`style`**", not "**`styles`**".
If you use "**`styles`**" it will be silently ignored.

test styling in both Android and iOS to verify consistent support

STYLING IMPORTS

- Components can import style objects
- Allows them to be shared between components

FLEXBOX ...

OBJECT COMPUTING

- Layout for all components with children is done with flexbox
  - supported by Yoga library (https://yogalayout.com/)
  - turned on by default
  - some differences from web usage
  - **`flex-direction`** default is **`column`** instead of **`row`**
  - **`flex-grow`** is just **`flex`**

FLEXBOX ...

- Review of properties that are the same as in CSS
  - **`justifyContent`** - **`flex-start`** (default), **`center`**, **`flex-end`**, **`space-around`**, **`space-between`**
  - **`alignItems`** - **`stretch`** (default), **`center`**, **`flex-start`**, **`flex-end`** applies to children
  - **`alignSelf`** - **`auto`** (default; uses **`alignItems`** value) and all **`alignItems`** values applies only to current element
  - **`flexWrap`** - **`nowrap`** (default; elements that don't fit can be clipped) and **`wrap`**

POSITION

- Only supported values for CSS **`position`** property are **`relative`** (default) and **`absolute`**
- Typically use flexbox and default **`position`** to position components rather than **`position: 'absolute'`**

BORDERS ...

- Defaults
  - `borderColor` -> `'black'`
  - `borderStyle` -> `'solid'`
    - as of 4/7/19, other values are currently rendered as solid on both Android and iOS
  - `borderWidth` -> `0`
- Can get solid black border by just setting `borderWidth`

... BORDERS

OBJECT COMPUTING

- When using **`borderRadius`**, **`backgroundColor`** goes outside border in corners unless **`overflow`** is set to **`hidden`**
- Example

```
myStyle: {
  backgroundColor: 'yellow',
  borderRadius: 10,
  overflow: 'hidden'
}
```

MARGIN AND PADDING

- No shortcut properties like in CSS
  - ex. cannot use **`margin: '10 20 30 40'`**
  - but React Native adds the style properties **`marginHorizontal`**, **`marginVertical`**, **`paddingHorizontal`**, and **`paddingVertical`**

FONT STYLE PROPERTIES

OBJECT COMPUTING

- **`fontFamily`**
  - only accepts a single font name, not multiple like in CSS
  - can specify a different **`fontFamily`** for each platform using **`Platform.OS`** (and a ternary) or **`Platform.select`**
- **`fontSize`**
  - defaults to **`14`**
- **`fontStyle`**
  - values are **`'normal'`** (default) and **`'italic'`**
- **`fontWeight`**
  - values are **`'normal'`** (same as **`400`**; default), **`'bold'`** (same as **`700`**), and strings containing the numbers **`100`** to **`900`** in increments of **`100`** (ex. **`'900'`**)

TEXT STYLE PROPERTIES ...

OBJECT COMPUTING

- **`textAlign`** What does `'auto'` do?
  - values are `'auto'` (default), `'center'`, `'left'`, `'right'`, and `'justify'` (only iOS)
- **`textDecorationLine`**
  - values are `'none'` (default), `'underline'`, `'line-through'`, and `'underline line-through'`
- **`textDecorationColor`** - a color
- **`textDecorationStyle`**
  - values are `'solid'` (default), `'dashed'`, `'dotted'`, and `'double'`

... TEXT STYLE PROPERTIES

OBJECT COMPUTING

- **`textShadowColor`** - a color
- **`textShadowOffset`**
  - value is an object with **`height`** and **`width`** properties
  - can be negative for shadows on top and left
- **`textShadowRadius`** - a number

To add a drop shadow to a component,
use Elevation styles on Android (not a great effect)
or ShadowPropTypesIOS styles on iOS (ignored on Android).

TRANSFORMS

OBJECT COMPUTING

- **`transform`** style property
  - value is an array of objects applied sequentially with no delay between
  - ex. **`transform: [{rotate: '90deg'}, {scale: 0.7}]`** one operation per object
- Supported operations
  - **`rotate`** (same as **`rotateZ`**), **`rotateX`**, **`rotateY`**, **`rotateZ`** values can be in degrees or radians; origin is original center of component
  - **`scale`**, **`scaleX`**, **`scaleY`** one use is to render thumbnails
  - **`translateX`**, **`translateY`** x values increase to right; y values increase going down; components can be translated out of view
  - **`skewX`**, **`skewY`**
  - **`perspective`** for 3D effects

`backfaceVisibility` style property can be used to create a "card effect" and has values **`'visible'`** and **`'hidden'`**.

PROVIDED FONTS

OBJECT COMPUTING

- Listed at https://github.com/react-native-training/react-native-fonts
- Android provides far fewer than iOS
- Android supports some generic font names, but iOS does not
  - `monospace`, `sans-serif`, `serif`

CUSTOM FONTS

- To use
  - create **`assets/fonts`** directory if it doesn't exist
  - download font files into this directory
  - follow steps on next two slides based on project type

IMPORTANT
Open font files and verify that the file names matches the font name exactly. In macOS, double-clicking a font file opens it in the **Font Book** app and displays its name in the title bar.
If the name differs, rename the file.

CUSTOM FONTS IN EXPO PROJECTS

- Do this in **`App.js`**
- Specify custom fonts in **`fontFamily`** style properties
- Fonts loaded this way do not currently support **`fontStyle`** or **`fontWeight`** properties, so load font variations separately

```
import {Font} from 'expo';
import React, {Component} from 'react';

export default class App extends Component {
  state = {fontLoaded: false};

  async componentDidMount() {
    await Font.loadAsync({
      'font name': require('./assets/fonts/font-file')
    });
    this.setState({fontLoaded: true});
  }

  render() {
    return this.state.fontLoaded ? <MyTopComponent /> : null;
  }
}
```

can't render components that use a font until it has been loaded

CUSTOM FONTS IN REACT NATIVE CLI PROJECTS

- Add to **`package.json`**

```
"rnpm": {
  "assets": ["./assets/fonts/"]
},
```

- Run **`react-native link`**
- Specify custom fonts in **`fontFamily`** style properties

DYNAMIC STYLES

- Since styles are defined in JavaScript code they can be dynamic
  - ex. can define light and dark theme style objects and allow users to switch between their use

STYLE FILES

- Styles can be defined in separate file from component definition
  - often named **`styles.js`** inside component-specific directory
  - export styles object that is imported where needed
  - allows sharing styles between components

most prefer to only put styles in a separate file when they are shared by multiple components