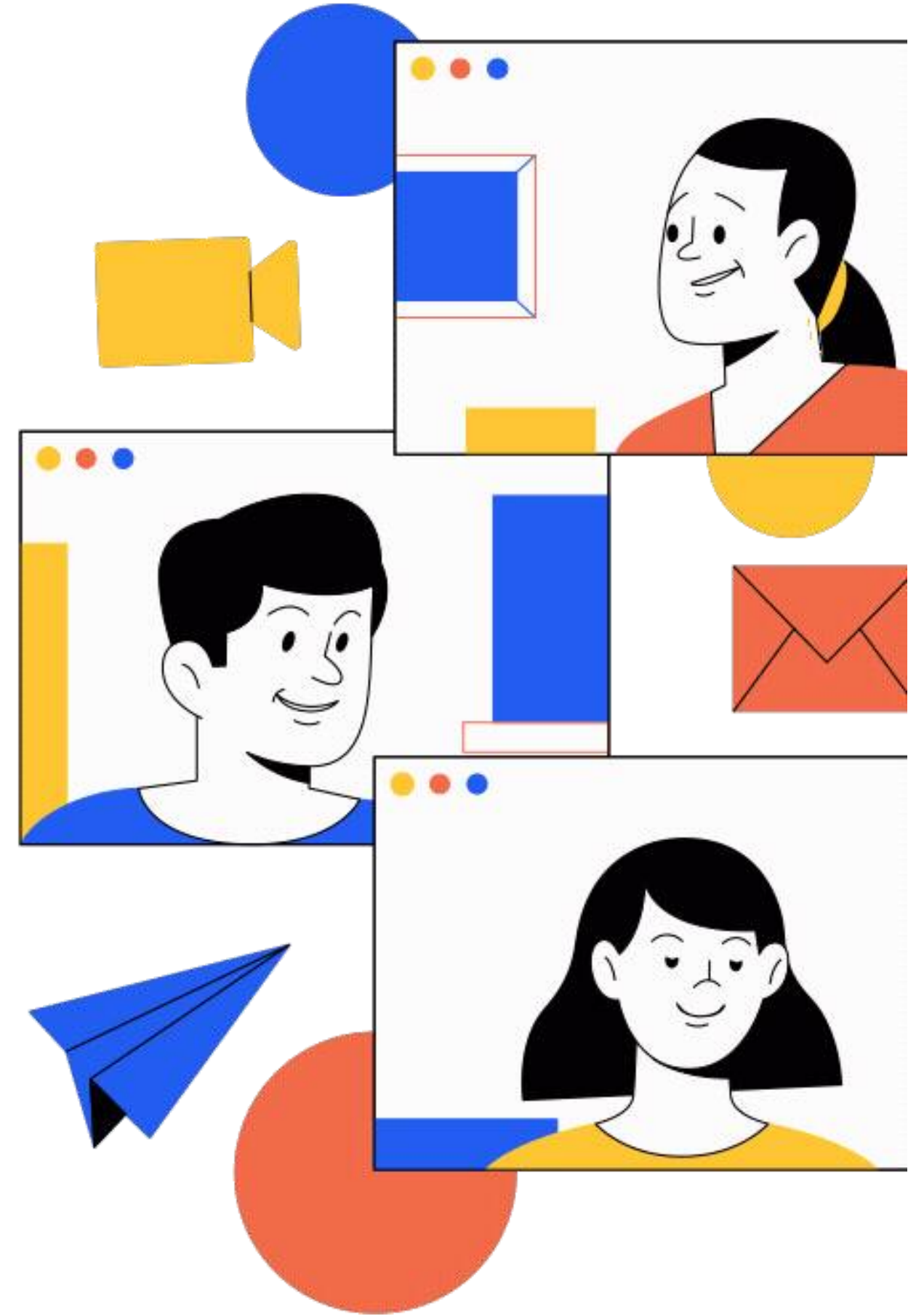


# Week 11 - React Native

How to style Mobile screens?

**Ahmed Fouad Lotfy**

React Session Lead



# Agenda



## What we'll cover in this session

- Conceptualizing Mobile Screens
- Creating Screens with React Native
- Using Flexbox for Mobile Layouts
- Styling Mobile Screens
- Live Demo

# How to Conceptualize Mobile Screens?

- **Identify user goals:**

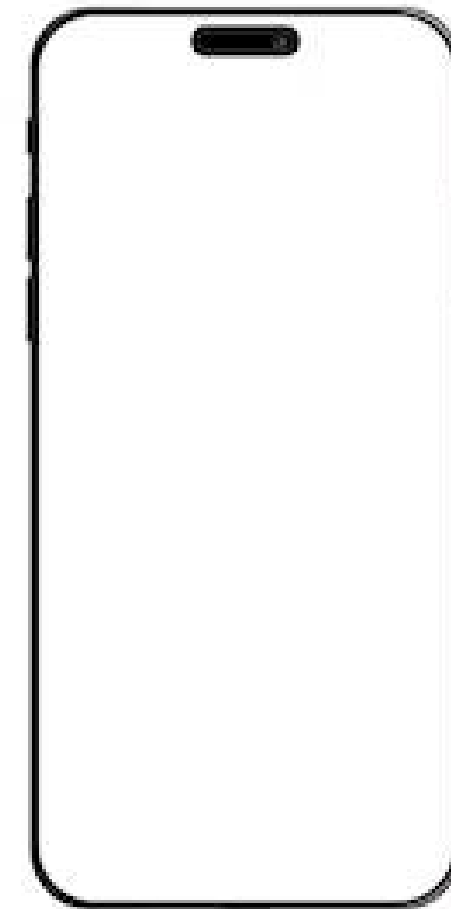
*What do users need to achieve on each screen?*

- **Define screen purpose:**

*Each screen should serve a distinct function.*

- **Keep navigation simple:**

*Ensure users can easily move between screens.*



# WEB



- **Layout Flexibility:** Highly flexible; grids, containers, sidebars, and various positioning methods can be used.
- **Interaction Types:** Mouse clicks, hover events, keyboard inputs, and touchscreen gestures for mobile-responsive sites.
- **Navigation Approach:** URL-based navigation, often using libraries like react-router for routing.
- **Design Flexibility:** Follows web standards, offering the freedom to design for any device but requires more attention to responsiveness (Ex: CSS)

# Mobile

- **Layout Flexibility:** More straightforward, focused on vertical stacking, optimized for smaller screens.
- **Interaction Types:** Touch and gesture-based (swiping, tapping, long presses) interactions dominate.
- **Navigation Approach:** Stack-based navigation (e.g., react-navigation), mimicking mobile app transitions.
- **Design Flexibility:** Follows platform-specific guidelines (Material Design for Android, HIG for iOS), ensuring native look and feel. (Ex: Stylesheets)

# How to create screens in React Native?

- **Screens in React Native are built using components:** You can define screens using `<View>`, `<Text>`, and other components. Each screen is rendered based on the app's navigation system.

```
import React,{useState} from 'react';
import { StyleSheet, Text, View,TextInput } from 'react-native';

return ( <View> <Text> My name is Gautham! </Text> </View>)
```

# Flexbox: Core Concepts

1. ***flexDirection***: Controls the direction in which items are laid out (row or column).  
*Default is column unlike in web the default is row*
2. ***justifyContent***: Aligns items along the main axis (horizontal or vertical).
3. ***alignItems***: Aligns items along the cross axis.

```
React Native

<View style={{ flexDirection: 'column', justifyContent: 'center',
alignItems: 'flex-start' }}>
  <Text>Flexbox Example</Text>
</View>
```



# Buttons Types

- 1. **TouchableOpacity:** General-purpose wrapper for touchable elements, allowing custom styling.
- 2. **Button:** Predefined button component with basic styling.
- 3. **Pressable:** More flexible touch handling, suitable for custom interactions.

Feature	TouchableOpacity	Button	Pressable
Customization	Highly customizable, allows for custom styles and child components.	Limited customization; styles are mostly defined by the default theme.	Highly customizable, with access to various states (pressed, hovered, etc.).
Feedback	Provides a fading effect on touch.	Displays a default press effect but has no fade.	Customizable feedback effects based on state changes.
Accessibility	Can be made accessible, but requires manual props (e.g., accessibilityLabel).	Built-in accessibility features that are automatically handled.	Supports accessibility features; requires proper props for full accessibility support.
Touch Handling	Handles onPress events and can support long press.	Only supports onPress.	Supports multiple touch events (onPress, onPressIn, onPressOut, onLongPress).
Performance	Lightweight and performs well for most use cases.	Efficient for basic button use cases.	Slightly more overhead due to additional state management capabilities.

# Best Practices for Mobile Layouts

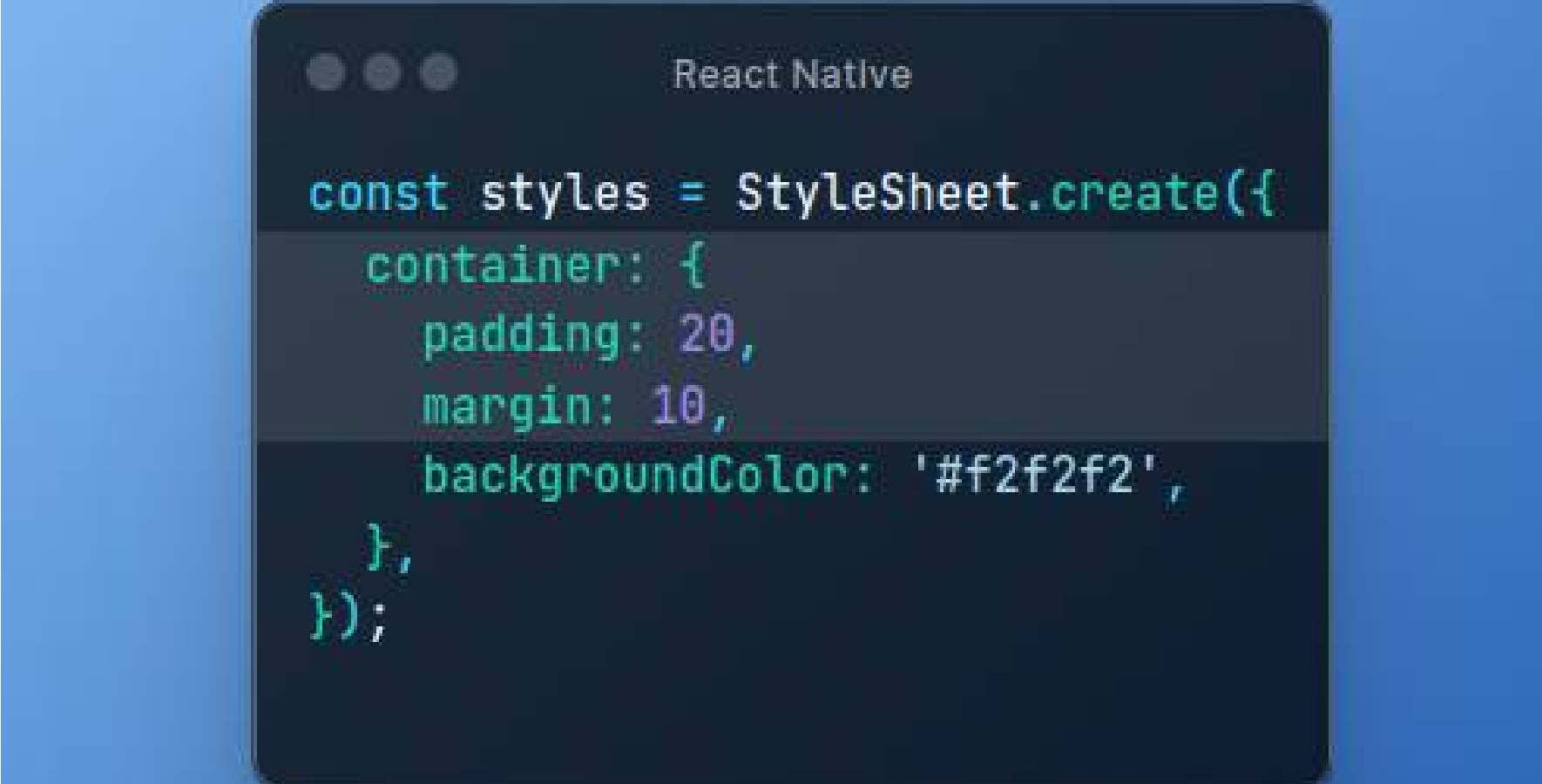
- **Layouts:** Use minimal and clean layouts for better readability.
  - Web:** Utilize traditional CSS box model with margin, padding, and border for layout management.
  - Mobile:** Emphasize minimal and clean layouts for better readability. Use Flexbox with a default column direction for layout efficiency.
- **Screens:** Use minimal and clean layouts for better readability.
  - Web:** Design responsive layouts using media queries for different screen widths.
  - Mobile:** Create adaptable designs for various screen sizes using percentage-based widths. Layouts should dynamically adjust, leveraging the Dimensions API for screen size.
- **Units of Measurement:** Design for different screen sizes using pixels-based widths.
  - Web:** Common units include px, em, rem, %, and vh/vw.
  - Mobile:** Primarily uses dp (density-independent pixels) for layout dimensions.



# Styling Mobile Screens

**Styles:** React Native uses inline styling or external stylesheets with the `StyleSheet.create` method. CSS-like properties: `margin`, `padding`, `backgroundColor`, etc.

1. **padding:** Adds space inside a component, between the content and its border.
2. **margin:** Adds space around the outside of a component.
3. **backgroundColor:** Sets the background color of a component.



```
const styles = StyleSheet.create({
  container: {
    padding: 20,
    margin: 10,
    backgroundColor: '#f2f2f2',
  },
});
```

# Responsive Mobile Design

**Responsive:** Responsive design adapts the UI to different screen sizes.

- Use percentages or Flexbox to create layouts that adjust dynamically.
- Best practices: Avoid fixed widths, use min/max dimensions.

```
React Native

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

const App = () => {
  return (
    <View style={styles.container}>
      <Text style={styles.header}>Welcome</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    padding: '5%',
    margin: '2%',
    backgroundColor: '#f2f2f2',
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  header: {
    fontSize: 24,
    fontWeight: 'bold',
    color: '#333',
    textAlign: 'center',
    width: '100%',
    maxWidth: 300,
  },
});

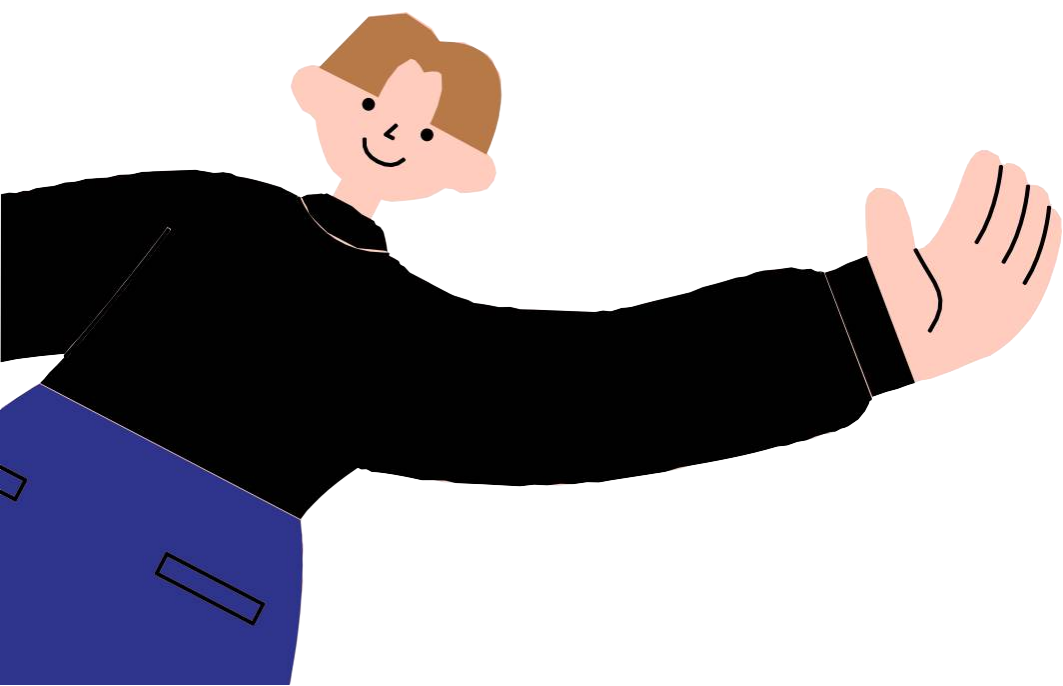
export default App;
```

**Demo  
Time**



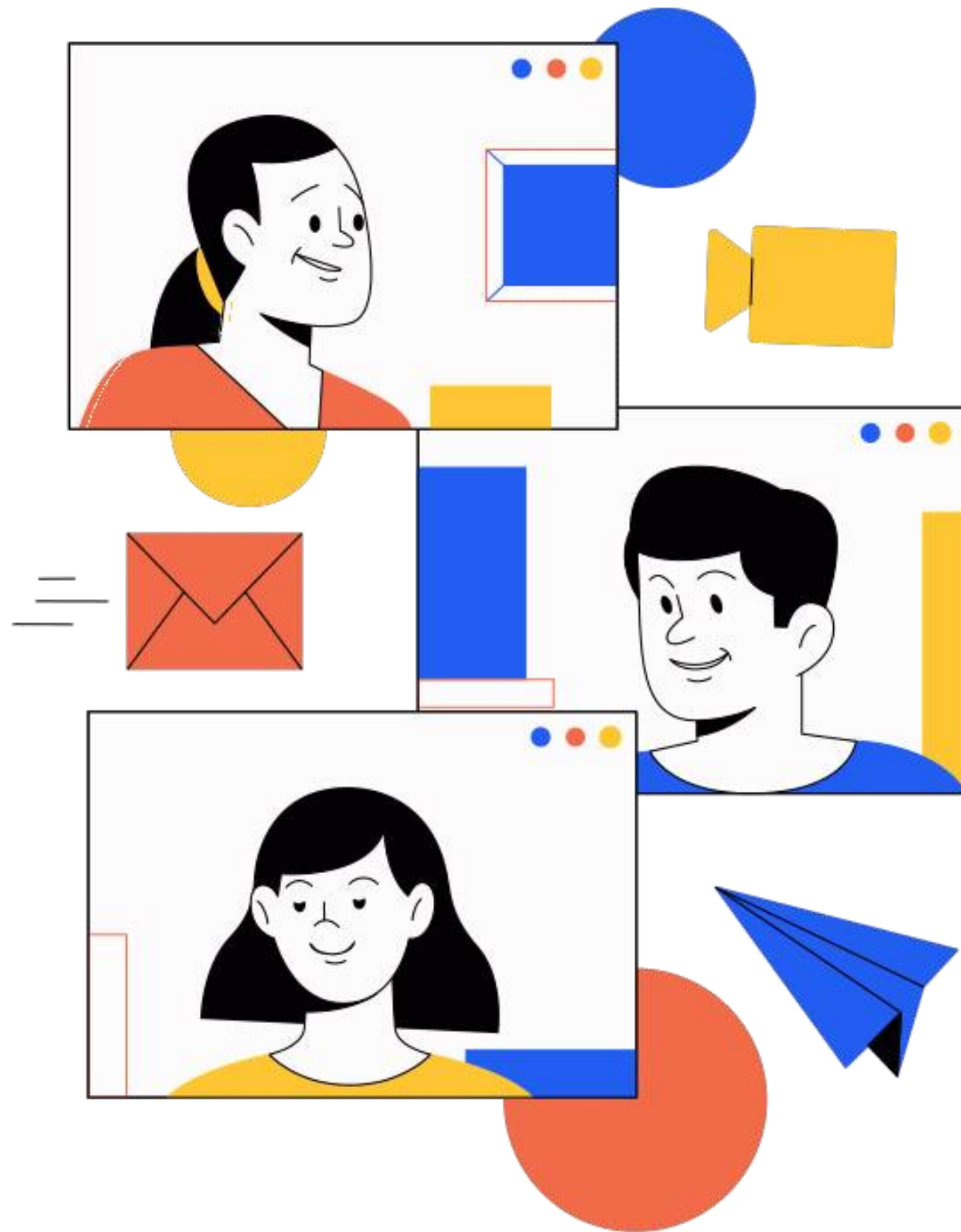


**Any Questions?**



# **Next Session: More React Native**





# Thank you for attending!

Feel free to email at [a.lotfy@fci-cu.edu.eg](mailto:a.lotfy@fci-cu.edu.eg) or reach me at circle anytime for any questions or clarifications!



Follow me on Github [@ahmeddx Fouad](https://github.com/ahmeddx Fouad)  
code and slides are found at this [github repo](#)