REACT NATIVE

Introduction

Naveen Pete

Agenda

- React
- Why React?
- React Native
- How does React Native work?
- Setting up Dev Environment
- Working with React Native Application
- Components
- JSX
- Registering Root Component
- Props
- State
- Style
- Navigation
- React Native Paper
- Q & A

React

- A JavaScript library for building UI
- Designed to solve some of the challenges and complexities in large-scale, data-driven web application development
- Developed and maintained by Facebook
- Released in 2013
- Current version – v16.8

Why React?

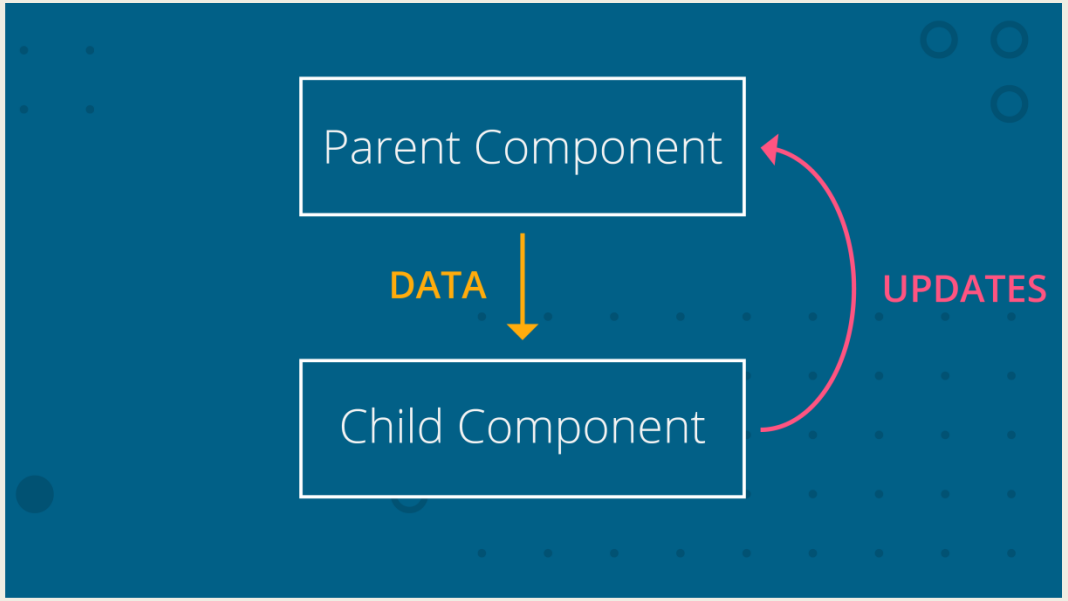
- Composition
  - *Combine simple functions to build complex functions*
  - *React builds up pieces of a UI using components*
  - *A Component*
    - is a key feature of React
    - encapsulates UI elements, data and the behavior of a view
    - allows you to break a complex web page into smaller, manageable & reusable parts
    - helps us create our own custom elements
    - groups many elements together and use them as if they were one element

Why React?

- React is Declarative
  - *Imperative Code*
    - Expressing a command; commanding
    - Instructs JavaScript on *HOW* it should perform each step
  - *Declarative Code*
    - We tell JavaScript *WHAT* we want to be done
    - Let JavaScript take care of performing the steps
- React is just JavaScript
  - *React builds on JavaScript*
  - *No need to learn new way of doing things*

Why React?

- Unidirectional Data flow
  - *Data lives in the parent component*
  - *Data flows from parent component to child component*
  - *Data updates are sent to the parent component*
  - *Parent performs the actual change*



Why React?

- Learn Once, Write Anywhere
  - *You can develop new features in React without rewriting existing code.*
  - *React can*
    - render on the server using Node
    - power mobile apps using React Native

React Native

- A framework for building native apps that run on iOS and Android devices
- Build apps using JavaScript and React
  - *Uses the same design philosophy as React*
  - *Lets you compose a rich mobile UI using declarative components*
- A React Native app is a real mobile app
  - *Uses the same fundamental UI building blocks as regular iOS and Android apps*
- Combines smoothly with components written in Swift, Java, or Objective-C
  - *It's easy to build part of your app in React Native, and part of your app using native code directly*

React Native

- Examples of some React Native apps
  - *Facebook*
    - Events Dashboard
  - *Instagram*
    - Push Notifications
    - Edit Profile
    - ‘Photos of’ view
  - *Skype*
  - *UberEATS*
  - *Pinterest*

Who’s using React Native?

`https://facebook.github.io/react-native/showcase`

How does React Native work?

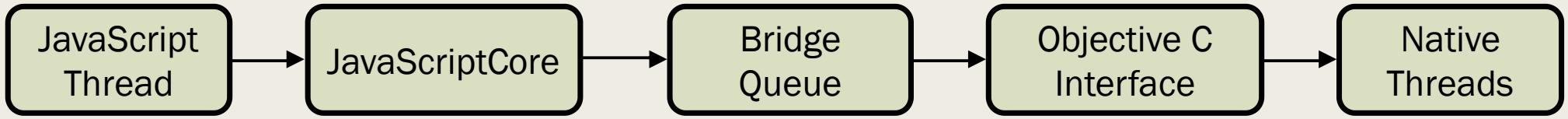
- React Native deals with 2 realms
  - *the JavaScript one*
  - *the Native one*
- Both are written in different technologies
- They communicate with each other using a ‘Bridge’. The communication is
  - *bidirectional*
  - *asynchronous*
- JS realm sends asynchronous JSON messages describing the action the native part is supposed to accomplish, the native side responds by performing the specified task(s)

How does React Native work?

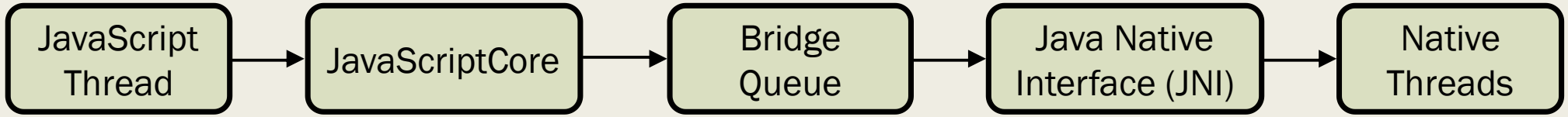
- Benefits
  - *Since it is asynchronous, it's non blocking, therefore allows for smooth view management on the screen*
  - *Since it is decoupled and based on interoperable languages, it's wide open to other frameworks and rendering systems*
- The Bridge
  - *plays the role of a Message Broker*
  - *is built in C/C++*
    - can be run on multiple platforms and OS
  - *embeds Apple's JavaScriptCore JS engine*
    - exposes API to access the JS engine capabilities
    - makes interoperability possible between JS and C/C++ code

How does React Native work?

- iOS



- Android



Setting up Dev Environment

- Android app dev environment on Windows
  1. *JDK v8 or newer*
     - https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
  2. *Python v2*
     - https://www.python.org/downloads/
  3. *Node v8.3 or newer*
     - https://nodejs.org/en/download/
  4. *Android Development Environment*
     - Android Studio
       - *https://developer.android.com/studio/index.html*
     - Install the Android SDK
       - *React Native build requires Android 9 (Pie)*
     - Configure the ANDROID_HOME environment variable
     - Add platform-tools to Path
  5. *React Native CLI*
     - npm install -g react-native-cli

Setting up Dev Environment

- iOS app dev environment on macOS
    1. *macOS v10 or newer*
    2. *Xcode – v9.4 or newer*
        - Install it from Mac App Store
    3. *Apple Developer Account*
        - https://developer.apple.com/programs/enroll
    4. *Homebrew*
        - https://brew.sh
    5. *Node v8.3 or newer*
        - brew install node
        - https://nodejs.org/en/download/
    6. *Watchman*
        - brew install watchman
    7. *React Native CLI*
        - sudo npm install -g react-native-cli

Working with React Native Application

- Creating a new application
  - *Open terminal window (or) command prompt*
  - *Use the React Native CLI to generate a new project called “HelloWorld"*

```
react-native init HelloWorld
```

- Running a React Native application
  - *iOS*

```
cd HelloWorld
react-native run-ios
```

    - You should see your new app running in the iOS Simulator shortly

Working with React Native Application

- Running a React Native application
  - *Android*
    - Prepare an Android virtual device
      - *Launch Android Studio*
      - *Open ‘./HelloWorld/android’ project folder*
      - *Click ‘AVD Manager’ tool on the toolbar. This opens up ‘Android Virtual Device Manager’ window*
        - If Android Studio is just installed, you will need to create a new AVD by clicking ‘Create Virtual Device…’ button
          - *Pick any phone from the list and clicke ‘Next’*
          - *Select Pie API Level 28 image*
          - *Click ‘Next’ and ‘Finish’ to create your AVD*
        - Click on the green triangle button next to your AVD to launch it
        - For more information on creating and managing AVDs visit this link
          - *https://developer.android.com/studio/run/managing-avds.html*

Working with React Native Application

- Running a React Native application (Continued)
  - *Android*
    - Run ‘react-native run-android’ inside your React Native project folder

```
cd HelloWorld
react-native run-android
```

    - You should see your new app running in your Android emulator shortly

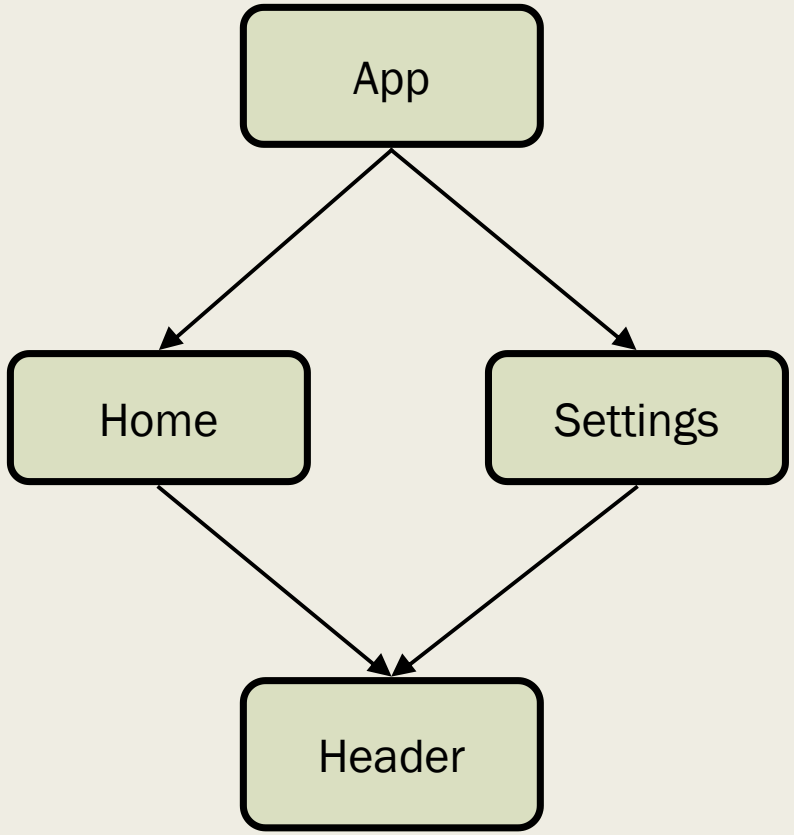
- Using a physical device
  - *To run the React Native app on a physical iPhone / Android device, follow the instructions given in the link below*
    - https://facebook.github.io/react-native/docs/running-on-device

Components

- Basic building block of React
- Encapsulate UI elements, data and the behavior of a view
- Allows you to break a complex screen into smaller, manageable & reusable parts
- Can be defined using
  - *a JS function*
    - Example - App component (./src/components/App.js)
  - *a JS class*
    - Example - Home component (./src/components/Home/index.js)
    - Should extend React.Component
    - Should contain render() method
- Can be user defined or built-in
  - *For built-in components check this URL*
    - https://facebook.github.io/react-native/docs/components-and-apis

Components

- Jokes application



JSX

- Syntax extension to JavaScript
- Lets us write JS code that looks like HTML
- More concise and easier to follow
- Should always return one root element

```
<View style={container}>
  <Header
    title="Home"
    subtitle="Best jokes!"
    type="home"
  />
  <View style={content}>
    <Text style={text}>{this.state.joke}</Text>
    <Button
      style={button}
      icon="sentiment-very-satisfied"
      mode="contained"
      onPress={() => this.getJoke()}
    >
      Another Joke
    </Button>
  </View>
</View>
```

Registering Root Component

- AppRegistry
  - *Is the JS entry point to running all React Native apps*
  - *App root components should register themselves with AppRegistry.registerComponent() method*
  - *Defined in ‘react-native’ library*

```
import { AppRegistry } from 'react-native';
import App from './src/components/App’;

AppRegistry.registerComponent('jokes', () => App);
```

Props

- A prop is any input that you pass to a React component
- Refer to attributes from parent components
- Added just like an HTML attribute
  - *prop name and value are added to the component*
- Props are stored on the ‘this.props’ object
- Props are read-only - a component must never modify its own props

```
// passing a prop to a component

<Welcome name='Hari' />
```

```
// access the prop inside the component
...
render() {
   return <h1>Hello, {this.props.name}</h1>
}
...
```

State

- Represents mutable data
- Affects what is rendered on the page
- Is managed internally by the component itself
- Is meant to change over time, commonly due to user input
- Is a plain JavaScript object that is used to record and react to user events

```
class User extends React.Component {
  state = {
    username: 'Hari'
  }

  render() {
    return (
      <div>Username: {this.state.username}</div>
    );
  }
}
```

State

- A component may update its state using ‘this.setState()’ method
- Whenever setState() is called, React, by default, re-renders the entire app and updates the UI

```
this.setState({
  subject: 'Hello! This is a new subject'
})
```

Style

- All of the core components accept a prop named style
- The style names and values usually match how CSS works on the web, except names are written using camel casing
  - *For e.g., 'backgroundColor' rather than 'background-color'*
- The style prop can be a plain old JavaScript object

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

const styles = StyleSheet.create({
  text: {
    fontSize: 25,
    textAlign: 'center',
  }
});

class Settings extends Component {
  ...
  render() {
    return (<View><Text style={styles.text}>Hello</Text></View>);
  }
}
```

It is cleaner to use StyleSheet.create() to define several styles in one place

Navigation

- React Navigation provides an easy to use navigation solution, with the ability to present common stack navigation and tabbed navigation patterns on both iOS and Android
- A standalone library that allows developers to set up the screens of an app with just a few lines of code
- Installing the package

```
npm install --save react-navigation
```

  - *https://reactnavigation.org/docs/en/getting-started.html*
- createStackNavigator(RouteConfigs, StackNavigatorConfig)
  - *Provides a way for your app to transition between screens where each new screen is placed on top of a stack*
- createAppContainer(Navigator)
  - *Containers are responsible for managing your app state and linking your top-level navigator to the app environment*

Navigation

```
import { createStackNavigator, createAppContainer } from "react-navigation";

import Home from './Home';
import Settings from './Settings';

const AppNavigator = createStackNavigator(
    {
        Home: Home,
        Settings: Settings
    }
);

export default createAppContainer(AppNavigator);
```

Navigation

- Navigation prop reference
  - *Each screen component in your app is provided with the navigation prop automatically*
  - *It contains various convenience functions that dispatch navigation actions on the route's router*
  - *It can be accessed using ‘this.props.navigation’*
  - *this.props.navigation*
    - navigate()
      - *we call the navigate function with the name of the route that we'd like to move the user to*
    - goBack()
      - *close active screen and move back in the stack*

Navigation

```
<Button
  title="Go to Home"
  onPress={() => this.props.navigation.navigate('Home')}
/>

<Button
  title="Go back"
  onPress={() => this.props.navigation.goBack()}
/>
```

React Native Paper

- Cross-platform Material Design for React Native
- A collection of customizable and production-ready components for React Native
- Follows Google’s Material Design guidelines
- Installing the package

```
npm install --save react-native-paper
npm install --save react-native-vector-icons
react-native link react-native-vector-icons
```

- Wrap your root component in Provider from react-native-paper

React Native Paper

```
import React from 'react';
import { AppRegistry } from 'react-native';
import { Provider as PaperProvider } from 'react-native-paper';

import App from './src/components/App';

const Main = () => {
   return (
      <PaperProvider>
         <App />
      </PaperProvider>
   );
};

AppRegistry.registerComponent('jokes', () => Main);
```

Reference

- React
  - https://reactjs.org/
- React Native
  - https://facebook.github.io/react-native/
- React Navigation
  - https://reactnavigation.org/en/
- React Native Paper
  - https://callstack.github.io/react-native-paper/index.html
- Useful links
  - *Understanding the React Native bridge concept* – Click here
  - *JavaScriptCore* – Click here
  - *JavaScript Promises, async/await* – Click here
  - *Material Design Icons* – Click here

Q & A

Thank you!