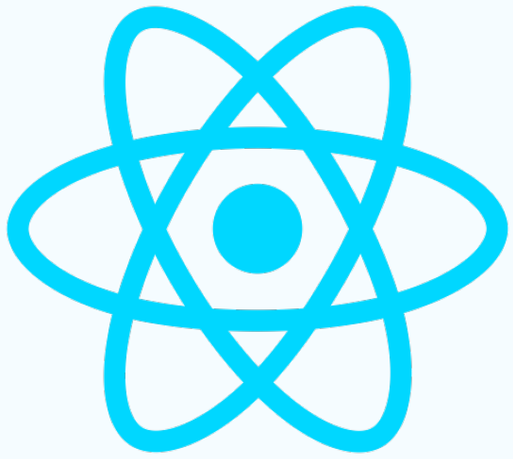

Intro to React Native

Asheville Women in Tech

What is React Native?

- React Native lets you build mobile apps with JavaScript and React instead of Swift, Kotlin or Java

- React Native apps render native UI elements for iOS and Android

React vs. React Native

- The core concepts of React like components, JSX, state, and props carry over to React Native
- React Native uses built-in native components instead of web components

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

// React
export const Hello = () => (
  <div>
    <p>Hello world!</p>
  </div>
);

// React native
export const HelloNative = () => (
  <View>
    <Text>Hello world!</Text>
  </View>
);
```

Getting Started

- There are two methods for starting a project: Expo and the React Native CLI

- Expo is like “create react app” for React Native. It will get your app up and running quickly

- The React Native CLI provides more flexibility and allows developers to access the native code

- You must have a Mac if you want to develop for iOS

JSX

- JSX looks like HTML, but it's actually an extension of JavaScript that produces React elements
- React does not require JSX but it is recommended
- Any valid JavaScript expression can go inside curly braces in JSX

```jsx
export const Avatar = ({
  image,
  username,
}) => (
  <View>
    <Image
      style={{
        width: 100,
        height: 100,
      }}
      source={{ uri: image }}
    />
    <Text>{username}</Text>
  </View>
);
```

Components

- Components are reusable pieces of code that are similar to JavaScript functions
- Components accept inputs called props and return React elements that render on the screen
- The two major types of components are stateful and stateless functional components

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

// Stateful
export class Post extends Component {
  state={
    description: 'This is a post!',
  }

  render() {
    const { description } = this.state;
    return (
      <Text>{description}</Text>
    );
  }
}

// Stateless functional component
export const PostStateless = ({
  description,
}) => (
  <Text>{description}</Text>
);
```

Props

- Two types of data control for components: props and state
- Props are set by the parent component
- Props are “read only” and should not be modified inside the component

```jsx
export const Fruit = (props) => {
  const { name } = props;
  return (
    <Text>{name}</Text>
  );
};

export const Fruits = () => (
  <>
    <View>
      <Fruit name="banana" />
    </View>
    <View>
      <Fruit name="apple" />
    </View>
  </>
);
```

State

- State should be used for data that will change
- State should not be modified directly
- When state changes, the component is re-rendered

```
export class Counter extends Component {
  state={
    count: 0,
  }

  increment = () => {
    this.setState(prevState => ({
      count: prevState.count + 1,
    }));
  }

  render() {
    const { count } = this.state;
    return (
      <>
        <Text>{count}</Text>
        <Button
          title="Click me!"
          onPress={() => this.increment()}
        />
      </>
    );
  }
}
```

Styling

- The style prop accepts a JavaScript object or an array of styles (the last style has precedence)
- The style names are the same as CSS, except that they are camel case
- Stylesheet.create() can be used to create styles
- React dimensions are unitless and represent density-independent pixels

```
const styles = StyleSheet.create({
  box: {
    width: 100,
    height: 100,
  },
  blue: {
    backgroundColor: 'blue',
  },
  red: {
    backgroundColor: 'red',
  },
});

export const Boxes = () => (
  <View>
    <View style={[styles.box, styles.blue]}>
      <Text>Blue</Text>
    </View>
    <View style={[styles.box, styles.red]}>
      <Text>Red</Text>
    </View>
  </View>
);
```

Flexbox

- Flexbox in React Native is the same as CSS, except the defaults are different

- flexDirection defaults to column instead of row

- flex only supports a single number

```
const styles = StyleSheet.create({
  avatarContainer: {
    display: 'flex',
    flexDirection: 'row',
    alignItems: 'center',
  },
  avatar: {
    width: 36,
    height: 36,
    borderRadius: 36 / 2,
  },
  avatarText: {
    marginLeft: 8,
    color: '#222',
  },
});

export const PostAvatar = ({ image, username }) => (
  <View style={styles.avatarContainer}>
    <Image
      style={styles.avatar}
      resizeMode="cover"
      source={{ uri: image }}
    />
    <Text style={styles.avatarText}>{username}</Text>
  </View>
);
```

View, Text and TextInput

- View is similar to a div. It is a container that supports flexbox, style, touch handling, and accessibility controls

- Text supports nesting, styling, and touch handling

- TextInput can be configured through props to support auto-correction, auto-capitalization, placeholder text, and keyboard types

```
export const Comment = ({
  comment,
  changeText,
}) => (
  <View>
    <Text>{comment}</Text>
    <TextInput
      placeholder="Add a comment..."
      onChangeText={changeText}
      value={comment}
    />
  </View>
);
```

Image and ImageBackground

- Image can display network images, static resources, temporary local images, and images from the camera roll
- Network images must be given a width and height
- ImageBackground accepts the same props as Image and children can be added. Width and height must be added

```
export const Icon = () => (
  <Image
    style={{
      width: 30,
      height: 30,
    }}
    source={require('../assets/pawprint.png')}
  />
);

export const Background = () => (
  <ImageBackground
    style={{ width: '100%', height: '100%' }}
    source={require('../assets/background.jpg')}
  >
    <Text>This goes on top</Text>
  </ImageBackground>
);
```

Buttons and Touchables

- React Native provides a basic button with default styling for both platforms

- Touchable components support tapping gestures and displaying feedback when a gesture is recognized

- Touchables include TouchableHighlight, TouchableNativeFeedback, TouchableOpacity, TouchableWithoutFeedback

```
export const PrimaryButton = ({
  handlePress,
}) => (
  <Button
    onPress={handlePress}
    title="Learn more"
    color="#1044b5"
    accessibilityLabel="Learn more"
  />
);

export const Action = ({
  handlePress,
}) => (
  <TouchableOpacity onPress={handlePress}>
    <Image
      style={{
        height: 30,
        width: 30,
      }}
      source={require('../assets/pawprint.png')}
    />
  </TouchableOpacity>
);
```

ScrollView

- ScrollView is a scrolling container that can host multiple components and views
- Must have a bounded height to work
- Renders all of its children at once
- ScrollView works best to display a small amount of things of a limited size

```
export const ScrollablePage = ({
  image,
}) => (
  <ScrollView>
    <Text>This is a ScrollView</Text>
    <Image
      style={{ height: 30, width: 30 }}
      source={{ uri: image }}
    />
    <Text>Keep on scrolling!</Text>
    <Image
      style={{ height: 30, width: 30 }}
      source={{ uri: image }}
    />
  </ScrollView>
);
```

FlatList

- FlatList renders items lazily so it saves memory and processing time

- FlatList supports horizontal mode, a header, a footer, separators, pull to refresh, scroll loading and multiple columns

- Requires two props: data and renderItem

```
export const LazyList = () => (
  <FlatList
    data={[
      { id: 'a123', name: 'Jeff' },
      { id: 'b456', name: 'Sarah' },
      { id: 'c789', name: 'Carlos' },
    ]}
    renderItem={({ item }) => (
      <Text>{item.name}</Text>
    )}
    keyExtractor={item => item.id}
  />
);
```

Resources

- React Native docs: facebook.github.io/react-native

- Presentation and Puppergram code: github.com/mmgolden/puppergram