Main Points/Key Points

Notes

Understanding React Native

1. **Basic React Native Component.**
    a. Refer to Code `Appendix A`.
    b. In React Native anything is displayed on the screen is a component.
    c. Considered component as a `class` with `extends` keyword and a `render()` function.
    d. Basic React Native Component:

Import

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

Explanation:

i. `import` keyword is used to load `react` module and assign it to a variable called `React`.
ii. This equivalent in importing packages/libraries in Java.
iii. It uses a de-structuring assignment to extract several object properties and assign to variables using a single statement.

Summary

Main Points/Key Points	Notes
	Understanding React Native

React Native Component

```
export default class App extends Component{
  render(){
    return (
      …
    );
  }
}
```

Explanation:

i. It defines a class, which inherit a `React` component.
ii. The `export default class` modifier makes the class public, which can be used/called in other files.
iii. `App extends Component {…}` is the basic building block for React Native UI.
iv. React Native Component contains `immutable` (`constant`) properties, `mutable(inconstant)` state variables and methods for rendering (`render()`).

Summary

Main Points/Key Points

Notes

Understanding React Native

JSX

```
<View style={styles.container}>
  <Text style={styles.header}>
   Welcome!
  </Text>
  <Text style={styles.contents}>
   This is React Native basic
   component.
  </Text>
</View>
```

Explanation:

i. JSX stands for **J**ava**S**cript e**X**tension. It mixes HTML-equivalent syntax into JavaScript code.
ii. Consider JSX as HTML tags or elements for React Native.
iii. JSX can be nested similar to HTML tags or element in web development.
iv. JSX is the element of the imported React Native Components or representation of the React Native Components (Objects).

Summary

Main Points/Key Points	Notes
	Understanding React Native

Styling

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'flex-start',
    alignItems: 'center',
    backgroundColor: '#F5FCFF',
  },
  header: {
    fontSize: 20,
    textAlign: 'center',
    margin: 10,
  },
  contents: {
    textAlign: 'center',
    color: '#333333',
    marginBottom: 5,
  },
});
```

Explanation:

i. React Native `StyleSheet` class is used to style the mobile app UI.
ii. The `StyleSheet` is similar to CSS used for web development.
iii. `App extends Component {…}` is the basic building block for React Native UI.
iv. React Native Component contains `immutable (constant)` properties, `mutable(inconstant)` state variables and methods for rendering (`render()`).

Summary

Main Points/Key Points

Notes

Understanding React Native

2. **React Native Entry File.**
    a. Refer to Code `Appendix B`.
    b. The entry point of React Native Application is `index.js`.
    c. It uses AppRegistry component to glue or bind App component (`App.js`) to the entry file (`index.js`)

index.js

```
import {AppRegistry} from 'react-native';
import App from './App';
import {name as appName} from './app.json';

AppRegistry.registerComponent(appName, ()
=> App);
```

Explanation:

i. It uses `AppRegistry` component to glue or bind App component (`App.js`) to the entry file (index.js).
ii. It also registers the name of the app to be installed on Android or iOS platform.
iii. `appName` is extracted from app description file (`app.json`), which created by `react-native init` command.

Summary

Main Points/Key Points

Notes

Understanding React Native

3. Managing Component Using State.

a. Data is created and managed in a component by using state.
b. State are declared when the component is created and it can be updated within the component by using a `setState()` function.
c. Another way to managed data is by using props or properties, which are passed down as parameters and they **CANNOT** be updated within components.
d. User interaction with components are good examples of how state works. For example, clicking buttons, checkboxes, filling forms, etc.

4. How to Manipulate Component State.

a. Refer to Code `Appendix C`.
b. State is a collection of values that a component manages and every time an UI changes using the `setState()` function, React Native will re-render the component.
c. Set initial state by using a `constructor()`.

Constructor

```
constructor(){
  super();
  this.state = {
      year: 2017,
      name: ‘Nader Dabit’,
      colors: [‘blue’]
  };
}
```

Summary

Main Points/Key Points

Notes

Understanding React Native

d. Change of state using `this.state` **DO NOT** work in React Native components.

Update State

```
 updateYear(){
  this.setState({year: 2018});
  //this.state.year = 2017; won't work
 }
}
```

e. State are changed through UI components like **Touchable** (`onPress`) and re-render when it is triggered.

Touchable

```
render() {
  return (
  <View>
  <Text>
   My name is: {this.state.name}
  </Text>
  <Text onPress={() => this.updateYear()}>
   The year is: {this.state.year}
  </Text>
  <Text>
   My colors are {this.state.colors[0]}
  </Text>
  </View>
  );
}
```

Summary

Main Points/Key Points

Notes

Introducing React Native

5. **Managing Component Using Props.**
    a. Props are short for properties and they are component's inherited values or properties from the parent component.
    b. Props' values can only be changed at the parent level, which they have been declared.
    c. Think props as *"a way of passing data from parent to the child."*
    d. The idea behind props is to allow developer to create a component that can be used in different places. In other words, props help developer to write reusable code.

6. **Static Props.**

User Defined Component

```
export default class App extends Component{
 render() {
   return (
    <BookDisplay book=”React Native in
     Action”/>
   );
 }
}

class BookDisplay extends Component {
  render() {
   return (
       <View>
         <Text>{this.props.book}</Text>
       </View>
       );
   }
 }
```

Summary

Main Points/Key Points	Notes

Introducing React Native

7. Dynamic Props.

User Defined Component

```
export default class App extends Component{
   constructor() {
    super();
     this.state = {
        book: 'React Native in Action'
     };
   }

   render() {
    return (
      <BookDisplay book={this.state.book}/>
    );
   }
}

class BookDisplay extends Component {
  render() {
   return (
   <View>
   <Text>{this.props.book}</Text>
   </View>
   );
  }
}
```

Summary

Main Points/Key Points

Notes

Introducing React Native

8. Updating Dynamic Props.

a. Refer to Code `Appendix D`.

b. Declare the state variable.

```
this.state = {book: 'React Native in Action'};
```

c. Write function that will update the state variable.

```
updateBook() {
   this.setState({book: 'Express in Action'});
}
```

d. Pass the function and the state down to child component as prop.

```
<BookDisplay updateBook={() => this.updateBook()}
 book={ this.state.book } />
```

e. Attach touch handler function in child component.

```
<Text onPress={ this.props.updateBook }>
```

Summary

Main Points/Key Points

Notes

Introducing React Native

9. **Props with Stateless Components.**
    a. Stateless components can only change props but not its state.
    b. It is useful when creating reusable components.

Stateless Props

```
const BookDisplay = (props) => {
  const { book, updateBook } = props
    return (
      <View>
      <Text onPress={ updateBook }>
       { book }
      </Text>
      </View>
    );
 }
```

10. **De-Structuring Props in a Stateless Component.**

Stateless Props

```
const BookDisplay = ({updateBook, book}) =>
{
 return (
      <View>
      <Text onPress={ updateBook }>
       { book }
      </Text>
      </View>
    );
}
```

Summary

Notes

Introducing React Native

11. References

a. Dabit, N. (2018). *React Native in Action*. New York, NY: Manning Publications Co.

b. Ravichandran, A. (2018). Props and State in React Native Explained in Simple English. Retrieved from https://codeburst.io/props-and-state-in-react-native-explained-in-simple-english-8ea73b1d224e

c. Facebook Inc. React Native Reference, version 0.57. Available at https://facebook.github.io/react-native/

Summary

Appendix A

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

export default class App extends Component {
  render() {
    return (
      <View style={styles.container}>
        <Text style={styles.header}>Welcome!</Text>
        <Text style={styles.contents}>This is React Native basic component.</Text>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'flex-start',
    alignItems: 'center',
    backgroundColor: '#F5FCFF',
  },
  header: {
    fontSize: 20,
    textAlign: 'center',
    margin: 10,
  },
  contents: {
    textAlign: 'center',
    color: '#333333',
    marginBottom: 5,
  }
});
```

Appendix B

```
import {AppRegistry} from 'react-native';
import App from './App';
import {name as appName} from './app.json';

AppRegistry.registerComponent(appName, () => App);
```

Appendix C

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

export default class App extends Component {
  constructor(){
    super();
      this.state = {
        year: 2016,
        name: 'Nader Dabit',
        colors: ['blue']
      }
  }

  updateYear(){
     this.setState({year: 2017});
     //this.state.year = 2017; won't work
  }

 render() {
   return (
      <View styles={styles.container}>
        <Text styles={styles.header}>React Native State</Text>
        <Text styles={styles.contents}>My name is: {this.state.name}</Text>
        <Text styles={styles.contents}onPress={() => this.updateYear()}>
          The year is: {this.state.year}
        </Text>
        <Text styles={styles.contents}>My colors are {this.state.colors[0]}</Text>
      </View>);
 }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'flex-start',
    alignItems: 'center',
    backgroundColor: '#F5FCFF',
  },
  header: {
    fontSize: 30,
    textAlign: 'center',
    margin: 10,
  },
  contents: {
    textAlign: 'center',
    color: '#333333',
    marginBottom: 5,
  }
});
```

Appendix D

```
export default class App extends Component {
  constructor(){
  super();
    this.state = {book: ‘React Native in Action’};
  }

  updateBook() {
    this.setState({book: ‘Express in Action’});
  }

  render() {
   return (
    <BookDisplay
     updateBook={ () => this.updateBook() }
     book={ this.state.book } />
    );
  }
}

class BookDisplay extends Component {
  render() {
   return (
    <View>
    <Text onPress={ this.props.updateBook }>
    {this.props.book}
    </Text>
    </View>
   )
  }
}
```