M Reza Faisal
Ilmu Komputer, FMIPA, ULM

Hello React Native

Instruksi

1. Membuat Project

Buat project dengan nama HelloWorld dengan perintah berikut. Harap diperhatikan saat menulis perintah ini di command prompt harus berada di drive dan folder tempat Anda menyimpan folder project HelloWorld ini.

```
npx react-native init HelloWorld
```

Buka folder HelloWorld pada Code Editor misal Visual Studio Code. Kemudian untuk pengguna sistem operasi Windows dan menggunakan emulator Android maka perlu dilakukan setup/konfigurasi tambahan pada folder yaitu:

- Tambahkan file local.properties pada folder android yang berada di dalam project HelloWorld.
- Tambahkan lokasi SDK Android pada file tersebut, berikut adalah contohnya.

```
sdk.dir = C:\\Android
```

2. Folder & File

Berikut adakah keterangan dari sebagian folder dan file yang terdapat pada project React Native.

Tabel 1. Keterangan Folder & File.

No	Nama Folder/File	Keterangan
1	`android/`	Direktori yang berisi kode native Android.
2	`ios/`	Direktori yang berisi kode native iOS.
3	`node_modules/`	Direktori ini berisi seluruh package Node.js dan dependencies yang diperlukan aplikasi dasar React Native. Namun kita dapat menambahkan package lain dengan cara menggunakan perintah "`npm install`"
4	`index.js`	File ini berisi daftar komponen yang digunakan pada saat aplikasi dijalankan. isi dari file ini dapat dilihat pada kode di bawah.

		`import {AppRegistry} from 'react-native';`
		`import App from './App';`
		`import {name as appName} from './app.json';`
		`AppRegistry.registerComponent(appName, () => App);`

		Aplikasi adalah komponen yang diekspor dari file App.js dan appName (baris kedua). appName adalah string yang berisi nilai aplikasi yang nilainya berada pada file app.json (baris ketiga) Baris keempat berfungsi untuk mendaftarkan komponen "App" kepada direktori yang sesuai dengan platform masing-masing.
5	`App.js`	File utama aplikasi untuk mendefinisikan komponen.
6	`app.json`	File berisi nama untuk aplikasi pada aplikasi Android dan iOS.

No	Nama Folder/File	Keterangan
7	`package.json`	File meta-data yang berisi informasi tentang project seperti nama project, nama pembuat, url project, package node yang diinstall pada project dan lain-lain.
8	`.babelrc`	File konfigurasi untuk package Babel. Babel adalah tool yang digunakan untuk mengkonversi kode ECMAScript 2015+ ke kode versi JavaScript versi yang dapat dikenali oleh browser atau lingkungan sekarang atau yang lebih tua.
9	`.buckconfig`	File konfigurasi untuk package Buck. Buck adalah sistem yang dibangun dan digunakan oleh Facebook yang berfungsi untuk mendukung pembuatan kode dan resource yang kecil dan dapat digunakan ulang serta mendukung bermacam bahasa pemrograman pada banyak platform.
10	`.flowconfig`	File konfigurasi untuk package Flow. Flow berfungsi untuk memeriksa kode JavaScript yang dapat membuat kode berjalan lebih cepat.
11	`.gitattributes`	File yang berisi atribut-atribut yang menunjuk ke pathname.
12	`.gitingnore`	File yang berisi nama dan tipe file/direktori yang tidak akan disimpan pada git repository jika project disimpan di sana.
	`.watchmanconfig`	File konfigurasi package watchman. Watchman berfungsi melakukan monitor file-file pada project jika terjadi perubahan maka akan dijalankan perintah atau fungsi yang dituliskan pada file ini.

3. Hello World

Pada sub bab ini akan diberikan beberapa hal dasar terkait pengembangan aplikasi dengan React Native. Hal yang pertama adalah saat mengembangkan aplikasi, jika kode aplikasi telah di-build dan di-deploy ke device/emulator maka ketika kode aplikasi diubah dan disimpan maka secara otomatis perubahan akan di-deploy ke device.

Ketika project dibuat maka file App.js telah berisi contoh kode. Deploy dan jalankan aplikasi project ini pada device Android dengan cara menuliskan perintah berikut pada command prompt ketika berada di folder project ini.

```
npx react-native run-android

atau

react-native run-android
```

Maka dapat dilihat tampilan aplikasi pada emulator seperti pada sebelah kiri di Gambar 1. Kemudian hapus isi kode pada App.js dan ganti dengan kode di bawah ini.

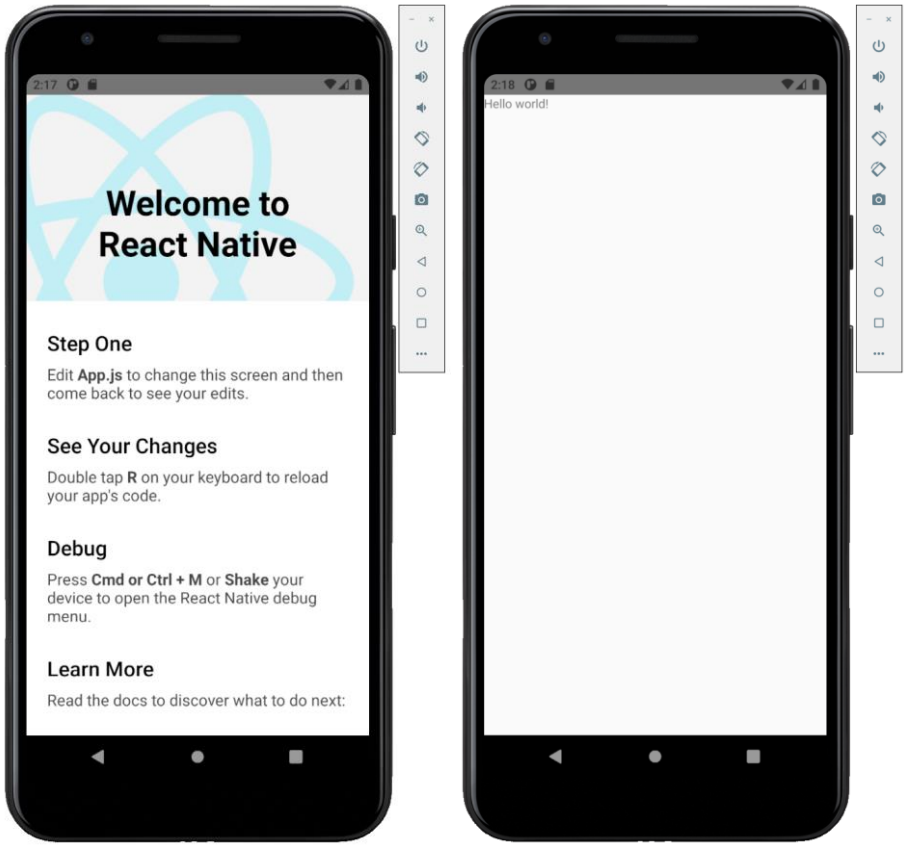
Kode 1. Hello World

App.js

```
1  import React from 'react';
2  import { Text, View } from 'react-native';
3
4  const HelloWorldApp = () => {
5    return (
```

```
6       <View>
7         <Text>Hello world!</Text>
8       </View>
9     )
10  }
11  export default HelloWorldApp;
```

Kemudian simpan file App.js, dan secara otomatis tampilan aplikasi pada device akan berubah seperti yang terlihat pada gambar sebelah kanan di Gambar 1.



Gambar 1. Antarmuka aplikasi HelloWorld.

Hal yang kedua adalah terkait dengan kode yang ditulis pada `App.js`. Seperti yang telah diterangkan di atas bahwa file `App.js` adalah file utama aplikasi. Berikut adakah keterangan setiap baris dari kode `App.js` di atas.

Baris 1, melakukan import `React` agar dapat menggunakan JSX yang berfungsi untuk mentransformasi menjadi komponen native masing-masing platform.

Baris 2, melakukan import komponen yang digunakan pada aplikasi yaitu `View` dan `Text` dari `react-native`.

Baris 4-10 adalah fungsi `HelloWorld` untuk mengembalikan komponen-komponen. Komponen yang dituliskan adalah `View` yang merupakan komponen untuk melakukan render container atau wadah. Sedangkan komponen `Text` berada di dalam wadah komponen `View`, komponen ini berfungsi untuk

melakukan render teks. Selain kedua komponen itu React Native masih memiliki banyak komponen lain yang dapat digunakan. Sedangkan HelloWorld sendiri adalah komponen baru yang kita buat.

Baris 6-8, Cara penulisan komponen dengan cara pada baris ini adalah sintaks JSX. JSX berfungsi untuk menggabungkan sintaks XML pada JavaScript sehingga kita dapat menulis markup language di dalam kode.

Seluruh kode `App.js` ditulis berdasarkan ES2015 (ES6) yang merupakan peningkatan dari bahasa JavaScript yang telah dikenal. ES6 sudah menjadi standar secara resmi tetapi belum digunakan pada semua web browser. Namun telah digunakan secara luas pada lingkungan selain web. Kata kunci `import`, `export`, `const` dan `from` adalah bagian dari ES6.

4. Style

Seperti halnya tag HTML, kita dapat memberikan style pada komponen. Berikut ada adalah contoh implementasi style pada komponen yang ditulis dengan JSX.

Kode 2. Inline Style

App.js

```
1  import React from 'react';
2  import { Text, View } from 'react-native';
3
4
5  const HelloWorldApp = () => {
6    return (
7      <View style={{flex: 1, justifyContent: "center", alignItems: "center"}}>
8        <Text style={{fontWeight: 'bold', fontSize: 32}}>Hello world!</Text>
9      </View>
10   )
11 }
12 export default HelloWorldApp;
14
```

Inline Style

Pada kode di atas, style ditulis secara inline artinya style langsung ditulis pada setiap komponen. Contohnya untuk style pada komponen `View` adalah sebagai berikut.

```
<View style={{flex: 1, justifyContent: "center", alignItems: "center"}}>
```

Sedangkan style untuk komponen `Text` adalah sebagai berikut.

```
<Text style={{fontWeight: 'bold', fontSize: 32}}>
```

Dari kedua contoh di atas dapat dibuat sintaks sebagai berikut.

```
<NAMA_KOMPONEN style={{atribut1: 'nilai_string', atribut2: nilai_angka}}>
</NAMA_KOMPONEN>
```

Internal Style

Cara pemberian style secara inline memiliki kekurangan karena jika ada lebih dari satu komponen harus diberikan style yang sama seperti contoh kode di bawah ini.

Kode 3. Inline Style 2

App.js

```
1  import React from 'react';
2  import { Text, View } from 'react-native';
4
5  const HelloWorldApp = () => {
6    return (
7      <View style={{flex: 1, justifyContent: "center", alignItems: "center"}
8  }>
9        <Text style={{fontWeight: 'bold', fontSize: 32}}>Hello world!</Text>
10       <Text style={{fontWeight: 'bold', fontSize: 32}}>Hello React Native!
11 </Text>
12     </View>
13   )
14 }
15 export default HelloWorldApp;
```

Pada komponen Text pada baris 9 dan 10 dapat dilihat kedua komponen ini diberikan style yang sama. Jika ada 10 komponen Text yang harus diberi style yang sama maka hal ini dapat membuat isi file menjadi lebih banyak dan ukuran file akan lebih besar.

Cara lain untuk memberikan style adalah secara internal seperti yang juga digunakan pada halaman web.

Kode 4. Internal Style

App.js

```
1  import React from 'react';
2  import { Text, View, StyleSheet } from 'react-native';
3
4  const styles = StyleSheet.create({
5    text_big: {
6      fontWeight: 'bold', fontSize: 23
7    },
8    container: {
9      flex: 1, justifyContent: "center", alignItems: "center"
10   }
11 })
12
13 const HelloWorldApp = () => {
14   return (
15     <View style={styles.container}>
16       <Text style={styles.text_big}>Hello world!</Text>
17       <Text style={styles.text_big}>Hello React Native!</Text>
18     </View>
19   )
20 }
21
22 export default HelloWorldApp;
```

Untuk membuat membuat style secara internal maka perlu ditambahkan class `StyleSheet` seperti terlihat pada baris ke-2. Kemudian pada baris ke-4 dapat dilihat dibuat konstanta `styles` yang menyimpan keluaran hasil dari method `StyleSheet.create()`. Kemudian setiap style diberi nama sebagai contoh `text_big` dan `container`.

Kemudian untuk memanggil style tersebut pada komponen dapat dilihat pada baris ke-15, ke-16 dan ke-17 dengan menggunakan konstanta `styles` diikuti nama style yang telah dibuat. Sebagai contoh untuk memanggil style `text_big` di dalam konstanta `styles` maka ditulis dengan cara berikut `styles.text_big`.

External Style

Cara yang lain membuat style dapat ditulis pada file terpisah dengan `App.js`. Sebagai contoh style-style disimpan pada file `AppStyle.js`.

Kode 5. AppStyle.js

AppStyle.js

```
1  import {StyleSheet} from 'react-native'
2
3  export default StyleSheet.create({
4      text_big: {
5          fontWeight: 'bold', fontSize: 40
6      },
7      container: {
8          flex: 1, justifyContent: "center", alignItems: "center"
9      }
10 });
```

Kemudian berikut adalah cara memanggil file `AppStyle.js` pada file `App.js`.

Kode 6. External Style

App.js

```
1  import React from 'react';
2  import { Text, View } from 'react-native';
3  import styles from './AppStyles'
4
5  const HelloWorldApp = () => {
6    return (
7      <View style={styles.container}>
8        <Text style={styles.text_big}>Hello world!</Text>
9        <Text style={styles.text_big}>Hello React Native!</Text>
10     </View>
11   )
12 }
13
14 export default HelloWorldApp;
```

Langkah pertama adalah memanggil file `AppStyle.js` pada baris ke-3 dengan menggunakan kata kunci `import` dikuti dengan nama variable `styles` dilanjutkan kata kunci from diikuti dengan `./AppStyle` yang menyatakan nama file `AppStyle.js`.

Setelah itu variable `styles` dapat digunakan pada komponen untuk memanggil style seperti yang terlihat pada baris ke-7, ke-8 dan ke-9.

5. Component

Core Component

Berikut ini adalah komponen-komponen ini React Native, yaitu:

a. Basic components
   - `View`, komponen kontainer.
   - `Text`, komponen untuk menampilkan teks.
   - `Image`, komponen untuk menampilkan gambar.
   - `TextInput`, komponen untuk input teks ke aplikasi melakui keyboard.
   - `ScrollView`, komponen yang berfungsi sebagai kontainer yang dapat diisi dengan komponen-komponen lain dan view.
   - `StyleSheet`, komponen untuk yang mempunyai fungsi seperti CSS.

b. User Interface
   - `Button`, komponen tombol untuk menangani input tekan.
   - `Switch`, komponen input boolean.

c. List Views
   - `FlatList`, komponen untuk menampilkan list yang dapat di-scroll.
   - `SectionList`, komponen seperti FlatList namun untuk list yang memiliki bagian/kelompok.

d. Android Components & API
   - `BackHandler`, komponen untuk mendeteksi ketika tombol hardware ditekan sebagai navigasi kembali.
   - `DrawerLayoutAndroid`, komponen untuk render DrawerLayout pada Android.
   - `PermissionsAndroid`, komponen untuk menyediakan akses ke model izin baru Android.
   - `ToastAndroid`, komponen untuk membuat alert Toast pada Android.

e. iOS Components & API
   - `ActionSheetIOS`, API untk menampilkan iOS action sheet atau share sheet.

f. Others
   - `ActivityIndicator`, komponen untuk menampilkan indikator loading.
   - `Alert`, komponen untuk menampilkan dialog alert dengan judul dan pesan.
   - `Animated`, library untuk membuat animasi.
   - `Dimensions`, interface untuk mendapatkan dimensi device.
   - `KeyboardAvoidingView`, komponen yang digunakan untuk mengatasi masalah yang terjadi pada view ketika virtual keyboard ditampilkan.
   - `Linking`, interface untuk berinteraksi dengan incoming dan outgoing app links.
   - `Modal`, komponen untuk menampilkan konten di atas view.
   - `PixelRatio`, komponen untuk mengakses pixel density dari device.
   - `RefreshControl`, komponen yang digunakan di dalam ScrollView untuk fitur refresh.

- `StatusBar`, komponen untuk mengatur status bar aplikasi.

Image

Sebagai contoh pada Kode 6 ditambahkan component `Image` sebagai berikut.

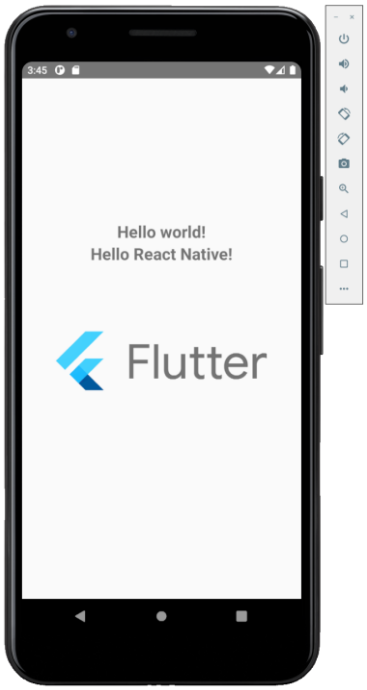
App.js

```
1   import React from 'react';
2   import { Text, View, Image } from 'react-native';
3   import styles from './AppStyles'
4
5   const HelloWorldApp = () => {
6     return (
7       <View style={styles.container}>
8         <Text style={styles.text_big}>Hello world!</Text>
9         <Text style={styles.text_big}>Hello React Native!</Text>
10        <Image source={require('./images/flutter-
11  logo.png')} style={{width: 300, resizeMode:'contain'}} />
12      </View>
13    )
14  }
15
16  export default HelloWorldApp;
```

Penggunaan component `Image` dapat dilihat dimulai pada baris ke-10. Sebelumnya telah disimpan gambar dengan nama file adalah `flutter-logo.png` pada folder `images`. Sehingga untuk mengakses file gambar lokal dapat dilihat caranya dengan memberikan nilai pada atribut `source` sebagai berikut.

```
require('./images/flutter-logo.png'
```

Hasilnya dapat dilihat pada Gambar 2.



Gambar 2. Contoh componen Image.

Diskusi

Selain menyimpan gambar pada lokal device, gambar juga dapat disimpan pada komputer server yang sehingga dapat diakses secara online.

- Sebutkan contoh implementasi penggunakan komponen `Image` untuk mengakses gambar secara online!
- Tulis kode implementasi komponen `Image` yang mengakses gambar secara online!

TextInput

Custom Component

Pada sub bab di atas adalah komponen-komponen yang telah disediakan oleh React Native dan dapat langsung digunakan. Jika komponen yang disediakan tidak mencukupi dengan kebutuhan maka dapat dibuat custom component. Custom component dapat dibuat dengan mengabungkan beberapa core component.

Pada Kode 7 adalah contoh pembuatan custom component secara internal, artinya custom component dibuat di dalam file yang sama dengan file `App.js`. Seperti yang disebutkan di banyak referensi tentang React Native, bahwa kode utama pada `App.js` yaitu kode pada baris ke-23 sampai ke-30 sebenarnya adalah component. Berdasarkan hal tersebut maka untuk membuat custom component dapat dilakukan dengan mengikuti kode tersebut.

Kode 7. Custom Componen Internal

App.js

```
1  import React from 'react';
2  import { Text, View, Image } from 'react-native';
3
4  const AppHeader = () => {
5    return (
6      <View style={{height:60, backgroundColor: 'gray'}}>
7        <Text style={{fontWeight: 'bold',fontSize: 32}}>Application Title</T
8  ext>
9      </View>
10   );
11 }
12
13 const AppContent = () => {
14   return (
15     <View>
16       <Text style={{fontWeight: 'bold',fontSize: 23}}>Content</Text>
17       <Image source={require('./images/flutter-
18 logo.png')} style={{width: 300, height:50, resizeMode:'contain'}} />
19     </View>
20   );
21 }
```

```
22
23  const HelloWorldApp = () => {
24    return (
25      <View>
26        <AppHeader/>
27        <AppContent/>
28      </View>
29    )
30  }
31
32  export default HelloWorldApp;
```

Pada Kode 7 dibuat dua buah custom component yaitu:

- `AppHeader` yaitu kode dimulai dari baris ke-4 sampai dengan baris ke-7.
- `AppContent` yaitu kode dimulai dari baris ke-13 sampai dengan baris ke-21.

Cara pembuatan komponen `AppHeader`, `AppContent` dan `HelloWorld` tersebut adalah mengikuti aturan pembuatan class pada JavaScript versi React Native atau dikenal dengan JavaScript ES6 (ECMAScript 6). JavaScript ES6 akan dibahas pada hands on labs selanjutnya. Isi dari class dapat berisi core component yang telah disebutkan pada sub bab sebelumnya.

Untuk menggunakan custom component yang telah dibuat dengan cara yang sama seperti menggunakan core component, yaitu dengan memanfaatkan JSX.

Custom component juga dapat ditulis pada file terpisah dengan file App.js atau menggunakan file external. Caranya dengan membuat file terpisah yang menyimpan kode dari baris ke-4 sampai ke-21. Sebagai contoh nama file yang digunakan adalah AppCustomComponents.js.

Kode 8. AppCustomComponents.js

AppCustomComponents.js

```
1   import React from 'react';
2   import { Text, View, Image } from 'react-native';
3
4   export const AppHeader = () => {
5       return (
6           <View style={{height:60, backgroundColor: 'gray'}}>
7           <Text style={{fontWeight: 'bold',fontSize: 32}}>App Title</Text>
8           </View>
9       );
10  }
11
12  export const AppContent = () => {
13      return (
14          <View>
15          <Text style={{fontWeight: 'bold',fontSize: 23}}>App Content</Text>
16          <Image source={require('./images/flutter-
17  logo.png')} style={{width: 300, height:50, resizeMode:'contain'}} />
18          </View>
19      );
```

```
20  }
```

Perbeda Kode 8 dengan Kode 7 adalah penggunaan kata kunci export pada baris ke-4 dan ke-12. Kata kunci ini digunakan agar class atau komponen dapat di akses dari file lain.

Selanjutnya dapat dilihat isi file App.js seperti yang dapat dilihat pada Kode 9. Untuk menggunakan custom component yang telah disimpan pada file AppCustomComponents.js maka perlu dilakukan import dengan cara seperti pada baris ke-3.

Kode 9. App.js dengan external custom component.

App.js

```
1   import React from 'react';
2   import { Text, View, Image } from 'react-native';
3   import {AppHeader, AppContent} from './AppCustomComponents'
4
5   const HelloWorldApp = () => {
6     return (
7       <View>
8         <AppHeader/>
9         <AppContent/>
10      </View>
11    )
12  }
13
14  export default HelloWorldApp;
```

Diskusi

Pada implementasinya custom compent dapat digunakan untuk banyak hal, misal component layout atau tata letak untuk header, footer atau menu. Custom component juga dapat digunakan untuk membuat tombol atau input yang mempunyai tampilan atau prilaku yang berbeda dengan core component.

- Sebutkan implementasi custom component yang lain!

Tugas

Lakukan tugas-tugas berikut:

1. Ikuti seluruh instruksi di atas.
2. Eksplorasi dan buat kode untuk implementasi setiap komponen yang disebutkan di atas.
3. Buat sebuah custom komponen dan gunakan pada aplikasi HelloWorld.

Referensi

1. shttps://reactjs.org/docs/hello-world.html
2. https://reactnative.dev/docs/components-and-apis
3. https://reactnative.dev/docs/stylesheet
4. https://medium.com/dailyjs/javascript-module-cheatsheet-7bd474f1d829

5. https://reactnative.dev/docs/intro-react