M Reza Faisal
Ilmu Komputer, FMIPA, ULM

React ES6

Instruksi

ES6 adalah singkatan dari ECMAScript 6. ECMAScript dikembangkan sebagai standar dari JavaScript dan ES6 adalah versi ke-6 dari ECMAScript yang dipublikasikan pada tahun 2015 sehingga kadang juga dikenal dengan nama ECMAScript 2015.

Fitur-fitur baru yang dimiliki oleh ES6 jika dibandingkan JavaScript versi sebelumnya adalah sebagai berikut:

- Class.
- Arrow function.
- Variable.
- Method .map() yang berfungsi mirip seperti array.
- Destructuring.
- Modules.
- Ternary operator
- Spread operator.

Pada hands on labs ini berisi pengenalan dasar-dasar JavaScript yang telah lama dikenal. JavaScript dan Java adalah berbeda. Kode program yang ditulis dengan JavaScript tidak perlu dikompilasi seperti halnya kode program yang ditulis dengan Java.

1. Statement & Komentar

Seperti kode program pada umumnya, program JavaScript juga merupakan serangkaian instruksi yang dikenal dengan istilah statement. Statement terdiri atas:

- Nilai (value).
- Operator.
- Ekspresi (expression).
- Kata kunci (keyword).
- Komentar (comment).

Statement-statement pada program dipisahkan oleh tanda titik koma (;), artinya sebuah statement diakhiri dengan tanda titik koma. Contohnya dapat dilihat pada Kode 1.

Kode 1. Contoh statement.

```
a = 13;         // Assign the value 13 to a
b = 23;         // Assign the value 23 to b
```

Pada kode di bawah ini juga dapat dilihat terdapa komentar yaitu pada teks berwarna hijau. Sebuah komentar diawali dengan tanda // kemudian diikuti dengan kalimat keterangan. Komentar juga dapat digunakan untuk menonaktifkan sebuah statement agar tidak dieksekusi.

2. Variable & Tipe Data

Variable adalah wadah untuk menyimpan nilai. Setiap wadah diberi nama agar mudah mengakses nilai didalamnya dengan cara memanggil nama wadah tersebut. Variable sebaiknya diberi nama dengan kata yang mencerminkan nilai disimpannya. Nama variable umumnya dimulai dengan huruf kecil. Jika variable terdiri atas dua suku kata maka kata awal ditulis dengan huruf kecil kemudian kata kedua dan seterusnya ditulis dengan huruf besar dan setiap kata disambung. Berikut adalah contohnya penamaan variable.

```
bilangan
bilanganPertama
namaDepan
fullName
variableDuaBelas
```

Kata kunci yang digunakan untuk mendeklarasikan variable adalah var. contohnya dapat dilihat pada kode di bawah ini.

```
var bil1 = 13;
var bil2 = 23;
var jumlah = bil1 + bil2;
```

Kata kunci var memiliki karakteristik sebagai berikut:

- Jika kata kunci var digunakan di luar fungsi maka variable tersebut bersifat global.
- Jika kata kunci var digunakan di dalam fungsi maka variable itu milik fungsi tersebut.
- Jika kata kunci var digunakan dalam blok percabangan atau perulangan maka variable tersebut masih tersedia diluar blok.
- Sehingga var dapat disimpulkan sebagai bahwa variable yang dibuat dengan var bersifat function scope bukan block scope.

Pada ES6 ditambahkan kata kunci let untuk membuat variable. Contoh penggunaannya seperti pada kode di bawah ini.

```
let x = 4;
```

Karakterisitik dari variable yang dibuat dengan kata kunci let adalah bahwa let bersifat block scope, artinya jika variable dibuat di dalam block seperti perulangan (looping) maka variable tersebut hanyar tersedia di dalam blok itu saja.

Kata kunci selanjutnya yang ditambahkan pada ES6 adalah const yang berfungsi untuk membuat kontanta, artinya nilai yang telah dimasukkan ke dalam variable tidak dapat diubah. Contohnya dapat dilihat pada kode di bawah ini.

```
const x = 11;
```

Seperti halnya variable yang dibuat dengan kata kunci let, maka variable ini juga block scope.

Variable pada JavaScript dapat diisi dengan nilai dari berbagai tipe data seperti angka, string atau obyek. Namun variable tidak perlu dideklarasikan tipe datanya terlebih dahulu sebelum variable diisi dengan nilai seperti yang dapat dilihat pada contoh-contoh di atas.

3. Operator

Salah satu contoh operator yang telah digunakan pada sub bab di atas adalah = (sama dengan) dan + (tambah). Operator-operator yang digunakan pada JavaScript adalah sama dengan operator-operator dalam bahasa pemrograman lain.

Berikut adalah jenis-jenis operator yang dimiliki JavaScript:

- Operator aritmatika seperti +, -, *, ** (untuk eksponensial, operator ini baru tersedia di ES6), /, %, ++ dan --.
- Operator assignmen seperti =, +=, -=, *=, /=, %= dan **=.
- Operator string yaitu +.
- Operator pembanding seperti ==, ===, !=, !==, <, >, >=, <= dan ?.
- Operator logika seperti &&, || dan !.
- Operator type seperti typeof dan instanceof.
- Operator bitwise seperti &, |, ~, ^, <<, >> dan >>>.

4. Function

Function adalah sub program atau kelompok kode yang dibuat untuk melakukan tugas tertentu. Sintaks membuat function pada JavaScript adalah sebagai berikut.

```
function nama_function(parameter1, parameter2, parameter3) {
  // statement
  // statement
}
```

Sintaks di atas adalah function yang tidak mengembalikan sebuah nilai, jika sebuah function mengambalikan sebuah nilai maka sintaksnya akan menjadi seperti berikut ini. Pada sintaks di bawah ini dapat dilihat cara untuk mengembalikan nilai dengan menggunakan kata kunci return.

```
function name_function (parameter1, parameter2, parameter3) {
  // statement
  // statement
  return value;
}
```

Pada Kode 2 adalah implementasi kedua tipe function di atas dalam aplikasi React Native.

Kode 2. AppJsExample.js

AppJsExample.js

```
1  import React from 'react';
2  import { Component } from 'react';
3  import { Text, View} from 'react-native';
4
5  function FunctionWithReturn(){
6      return "Function With Return";
7  }
8
9  var string1 = "";
10 function FunctionWithOutReturn(value) {
11     string1 = value;
12 }
```

```
13
14  function AppJsExample() {
15      var string2 = FunctionWithReturn();
16      FunctionWithOutReturn("Function With Out Return");
17
18      return (
19          <View>
20              <Text>Hello JavaScript ES6</Text>
21              <Text>{string1}</Text>
22              <Text>{string2}</Text>
23          </View>
24      )
25  }
26
27  export default AppJsExample;
```

Contoh function yang tidak mengembalikan nilai dapat dilihat pada baris ke-10 sampai baris ke-12 dengan nama function adalah FunctionWithOutReturn(). Sedangkan contoh function yang mengembalikan nilai adalah seperti yang dapat dilihat pada baris ke-5 sampai dengan ke-7 dengan nama function adalah FunctionWithReturn(). Selain itu function yang mengembalikan nilai juga dapat dilihat pada function dengan nama AppJsExample().

Untuk function FunctionWithReturn(), function yang mengembalikan nilai umumnya ada variable yang menampung keluaran dari function tersebut. Hal tersebut dapat dilihat pada baris ke-15. Variable yang digunakan adalah untuk menampung adalah string2. Selanjutnya untuk menampilkan nilai variable tersebut pada aplikasi dapat dilihat pada baris ke-22.

Pada baris ke-9 dapat dilihat deklarasi variable dengan nama string1 dengan inisial nilai adalah “”. Sebagaimana yang dijelaskan pada sub bab tentang Variable & Tipe Data bahwa variable yang dideklarasikan dengan menggunakan kata kunci var diluar function maka sifatnya adalah global, artinya nilainya dapat diupdate dan diakses dimana saja. Cara mengupdate variable string1 ini dilakukan dengan menggunakan function FunctionWithOutReturn(). Kemudian untuk menampilkannya pada aplikasi dapat dilakukan dengan cara seperti pada baris ke-21.

5. Arrow Function

Array function adalah cara yang diberikan pada ES6 untuk membuat penulisan function lebih singkat. Sintaks arrow function dapat dilihat di bawah ini.

```
nama_function = (parameter1, parameter2, parameter3) => {
    // statement
}
```

Jika function hanya memiliki sebuah statement saja dan hanya untuk mengembalikan value maka dapat ditulis seperti berikut ini.

```
nama_function = () => value;
```

Jika Kode 2 diubah menjadi dengan menggunakan cara penulisan arrow function maka hasilnya dapat dilihat pada Kode 3.

Kode 3. AppJsExample.js dengan arrow function.

AppJsExample.js

```
1   import React from 'react';
2   import { Text, View } from 'react-native';
3
4   const FunctionWithReturn = () => {
5       return "Function With Return";
6   }
7
8   var string1 = "";
9   const FunctionWithOutReturn = (value) => {
10      string1 = value;
11  }
12
13  const  AppJsExample = () => {
14      var string2 = FunctionWithReturn();
15      FunctionWithOutReturn("Function With Out Return");
16
17      return (
18          <View>
19              <Text>Hello JavaScript ES6</Text>
20              <Text>{string1}</Text>
21              <Text>{string2}</Text>
22          </View>
23      )
24  }
25
26  export default AppJsExample;
```

6. Class

Class adalah jenis dari function, tetapi pembuatannya menggunakan kata kunci class dengan method constructor() yang dapat disematkan didalamnya. Sintaks pembuatan class dapat dilihat pada kode di bawah ini.

```
class NamaClass {

}
```

Penamaan class umumnya dimulai dengan huruf besar. Jika nama class terdiri atas dua atau lebih suku kata maka suku kata kedua dan berikutnya ditulis dengan huruf besar. Sebagai contoh dapat dilihat pada kode berikut ini.

```
class Person {

}
```

```
class DatabaseConnector {

}
```

Pada konsep pemrograman berbasis obyek (PBO) sebuah class diibaratkan sebagai cetak biru atau gambar desain. Class diibaratkan sebagai bukan benda nyata dan belum bisa digunakan. Untuk menggunakan class terlebih dahulu mengubah class menjadi obyek. Obyek dapat diibaratkan sebagai benda nyata yang dibuat berdasarkan spesifikasi dari class.

Sebuah class dapat memiliki:

- Constructor berfungsi untuk menentukan bagaimana obyek dibuat atau diinstansiasi.
- Property berfungsi sebagai variable.
- Method adalah fungsi untuk melakukan sesuatu.

Sifat-sifat (property dan method) sebuah class dapat diturunkan ke class lainnya. Artinya sebuah class dapat memiliki induk atau orang tua. Pada PBO, hal ini dikenal dengan istilah inheritance. Berikut adalah sintaks yang digunakan hal tersebut adalah sebagai berikut.

```
class Induk {

}

class Anak extends Induk {

}
```

Pada kode di atas dapat dilihat terdapat class dengan nama Induk. Kemudian dibuat class baru dengan nama Anak, untuk mendapat sifat-sifat dari class Induk maka class Anak dapat menggunakan kata kunci `extends` seperti pada kode di atas.

Pada Kode 4 ini adalah contoh penggunaan class pada Reach Native.

Kode 4. AppClassExample.js

AppClassExample.js

```
1  import React from 'react';
2  import { Component } from 'react';
3  import { Text, View } from 'react-native';
4
5  class Manusia {
6      gender = "";
7      tinggi = 0;
8      berat = 0;
9      nama = "";
10
11     constructor(gender){
12         this.gender = gender;
13     }
14
15     Speak() {
16         return this.nama;
17     }
18 }
19
20 class AppClassExample extends Component {
21     render() {
22         const org = new Manusia("pria");
23         org.tinggi = 172;
24         org.nama = "Budi";
25         var namaOrg = org.Speak();
26
27         return (
28             <View>
29                 <Text>Contoh Penggunaan Class</Text>
30                 <Text>{org.tinggi}</Text>
31                 <Text>{namaOrg}</Text>
32             </View>
33         );
```

```
34      }
35  }
36
37  export default AppClassExample;
```

Untuk menggunakan file ini agar menjadi file utama aplikasi maka modifikasi file index.js menjadi seperti berikut ini.

index.js

```
/**
 * @format
 */

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

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

Pada Kode 4 dapat dilihat terdapat class dengan nama Manusia. Class ini memiliki sebuah constructor, empat property dan satu method. Contoh penulisan constructor dapat dilihat pada baris ke-11. Property dapat dilihat pada baris ke-6 sampai dengan baris ke-9. Sedangkan contoh method dapat dilihat pada baris ke-15.

Kemudian contoh inheritance dapat dilihat pada baris ke-20, di sini dibuat class dengan nama AppClassExample, agar class ini menjadi komponen maka class ini harus menjadi anak dari class Component. Selanjutnya pada class terdapat method render() yang digunakan untuk menulis JSX atau hal-hal lain yang ingin ditampilkan pada antarmuka aplikasi.

Kemudian pada baris ke-22 dapat dilihat contoh pembuatan obyek dengan nama org yang dibuat dari class Manusia. Untuk membuat obyek diberikan nilai “pria” karena hal ini sesuai dengan constructor yang ada pada class tersebut. Kemudian setelah obyek org dibuat dapat dilihat cara mengakses dan memberi nilai pada property seperti pada baris ke-23 dan ke-24. Dan contoh untuk mengakses method dapat dilihat pada baris ke-25. Selanjutnya untuk menampilkan nilai ke layar aplikasi dapat dilihat pada baris ke-30 dan ke-31.

Contoh berikutnya adalah memanfaatkan class untuk membuat custom component.

Kode 5. AppCustomComponentsClass.js

AppCustomComponentsClass.js

```
1   import React from 'react';
2   import { Component } from 'react';
3   import { Text, View, Image } from 'react-native';
4
5   class AppHeader extends Component {
6       render() {
7           return (
8               <View style={{height:60, backgroundColor: 'gray'}}>
9                   <Text style={{fontWeight: 'bold',fontSize: 32}}>App Title Class</Text>
10              </View>
11          );
12      }
13  }
14
15  class AppContent extends Component {
16      render() {
```

```
17          return (
18              <View>
19                  <Text style={{fontWeight: 'bold',fontSize: 23}}>App Content Class</Text>
20                <Image  source={require('./images/flutter-logo.png')}  style={{width:  300,
21  height:50, resizeMode:'contain'}} />
22              </View>
23          );
24      }
25  }
26
27  export {AppHeader, AppContent}
```

Selanjutnya untuk menggunakan custom component di atas tidak ada perbedaannya dibandingkan dengan custom component yang dibuat arrow function. Pada Kode 6 adalah file utama aplikasi untuk menggunakan custom component dari Kode 5.

Kode 6. AppClass.js.

AppClass.js

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

Dan pada Kode 7 adalah isi file index.js untuk menggunakan AppClass.js.

Kode 7. index.js untuk AppClass.js.

index.js

```
/**
 * @format
 */

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

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

7. Array Method

JavaScript memiliki beberapa method untuk mengelola array. Salah satu method yang paling bermanfaat adalah .map(). Method ini memungkinkan kita untuk melakukan iterasi pada array dan memanipulasi item-item didalamnya.

Berikut adalah contoh data yang disimpan pada array.

```
const buah = ['apple', 'banana', 'orange'];
```

Dan berikut adalah contoh penggunaan method .map() untuk mengakses array di atas pada aplikasi atau komponen.

Kode 8. AppArrayMethod.js

AppArrayMethod.js

```
1  import React from 'react';
2  import { Component } from 'react';
3  import { Text, View } from 'react-native';
4
5  class AppArrayMethod extends Component {
6      render() {
7          const buah = ['apple', 'banana', 'orange'];
8
9          return (
10             <View>
11                 <Text>Contoh Penggunaan Array Method</Text>
12                 { buah.map((item) => (
13                 <Text key={item}>{item}</Text>
14                 ))}
15             </View>
16         );
17     }
18 }
19
20 export default AppArrayMethod;
```

Dan berikut adalah index.js yang digunakan untuk menggunakan komponen AppArrayMethod di atas.

Kode 9. index.js untuk AppArrayMethod.js

index.js

```
/**
 * @format
 */

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

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

Array juga dapat dibuat memiliki atribut seperti id dan value. Berikut adalah contoh array dengan kedua atribut tersebut.

Kode 10. AppArrayMethod.js dengan data array dengan atribut id dan value.

AppArrayMethod.js

```
1  import React from 'react';
2  import { Component } from 'react';
3  import { Text, View } from 'react-native';
4
5  class AppArrayMethod extends Component {
6      render() {
7          const buah = [
8                        {"id":1, "value":'Apple'},
9                        {"id":2, "value":'Banana'},
10                       {"id":3, "value":'Orange'},
11                     ];
```

```
12
13          return (
14              <View>
15                  <Text>Contoh Penggunaan Array Method</Text>
16                  { buah.map((item) => (
17                  <Text key={item.id}>{item.value}</Text>
18                  ))}
19              </View>
20          );
21      }
22  }
23
24  export default AppArrayMethod;
```

8. Module

Module JavaScript digunakan dengan tujuan untuk memecah-mecah kode program menjadi beberapa file. Hal ini membuat untuk memudahkan untuk mengelola kode program. Untuk implementasi module digunakan kata kunci `import` dan `export`.

export

Kata kunci export berfungsi untuk meng-export class, function atau variable dari file. Ada dua tipe export yaitu:

- Named.
- Default.

Untuk menggunakan kata kunci export tipe named maka dalam sebuah file dapat berisi beberapa class atau function. Contohnya dapat dilihat pada Kode 5. Tipe export ini dapat meng-export beberapa class sekaligus.

import

Untuk meng-import module dengan dua cara berdasarkan tipe export yang digunakan. Jika custom component di-export dengan tipe named maka cara yang digunakan untuk import adalah sebagai berikut atau dapat dilihat pada Kode 6 baris ke-3.

```
import {AppHeader, AppContent} from './AppCustomComponentsClass'
```

Jika class atau function di-export dengant tipe default maka cara tidak perlu digunakan tanda kurung kurawal ({ }).

Berikut adalah contoh implementasi export tipe default. Cara export ini dapat dilihat pada Kode 11 baris ke-15.

Kode 11. AppCustomComponentFooter.js

```
AppCustomComponentFooter.js
1   import React from 'react';
2   import { Component } from 'react';
3   import { Text, View } from 'react-native';
```

```
4
5   class AppCustomComponentFooter extends Component {
6       render() {
7           return (
8               <View style={{height:60, backgroundColor: 'gray'}}>
9                   <Text style={{fontWeight: 'bold',fontSize: 32}}>App Footer Class</Text>
10              </View>
11          );
12      }
13  }
14
15  export default AppCustomComponentFooter;
```

Jika custom component ini digunakan pada Kode 6 maka cara import-nya dapat dilihat pada Kode 12 baris ke-4.

Kode 12. AppClass.js dengan import default.

AppClass.js

```
1   import React from 'react';
2   import { Text, View, Image } from 'react-native';
3   import {AppHeader, AppContent} from './AppCustomComponentsClass'
4   import AppCustomComponentFooter from './AppCustomComponentFooter'
5
6   const AppClass = () => {
7     return (
8       <View>
9         <AppHeader/>
10        <AppContent/>
11        <AppCustomComponentFooter/>
12      </View>
13    )
14  }
15
16  export default AppClass;
```

9. Destructuring

Destructuring adalah cara yang lebih mudah untuk mendapatkan nilai dari sebuah array atau obyek.

Destructuring Array

Untuk mengakses nilai pada array umumnya digunakan dengan cara berikut ini.

```
const siswa = ['budi', 'wati', 'iwan'];

const siswa1 = siswa[0];
const siswa2 = siswa[1];
const siswa3 = siswa[2];
```

Namun dengan destructuring maka cara di atas dapat diganti dengan kode berikut ini.

```
const siswa = ['budi', 'wati', 'iwan'];
const [siswa1, siswa2, siswa3] = siswa;
```

Dapat dilihat untuk mengisi nilai variable dapat dilakukan sekaligus dengan kode yang lebih singkat. Pada Kode 13 dapat dilihat contoh penggunaannya pada aplikasi.

Kode 13. AppDestructuringArray.js

AppDestructuringArray.js

```
1  import React from 'react';
2  import { Text, View } from 'react-native';
3  import { NavigationContainer } from '@react-navigation/native';
4
5  const AppDestructuringArray = () => {
6      const siswa = ['budi', 'wati', 'iwan'];
7      const [siswa1, siswa2, siswa3] = siswa;
8
9      return (
10         <View>
11             <Text>Daftar Siswa</Text>
12             <Text>{siswa1}</Text>
13             <Text>{siswa2}</Text>
14             <Text>{siswa3}</Text>
15         </View>
16     )
17 }
18
19 export default AppDestructuringArray;
```

Pada baris ke-6 dapat dilihat deklarasi data yang disimpan pada variable siswa. Selanjutnya untuk mengambil setiap nilai dan disimpan ke variable siswa1, siswa2 dan siswa3 dapat dilihat caranya pada baris ke-7. Kemudian untuk menampilkannya pada antarmuka aplikasi dapat dilihat pada baris ke-12 sampai dengan baris ke-14.

Misal nilai yang diambil hanya data ke-3 saja maka kode di atas dapat dimodifikasi menjadi potongan kode berikut ini. Pada baris ke-2 selanjutnya pada baris ke-7 dapat dilihat cara untuk menampilkan ke antarmuka aplikasi.

```
1  const siswa = ['budi', 'wati', 'iwan'];
2  const [, , siswa3] = siswa;
3
4  return (
5      <View>
6          <Text>Daftar Siswa</Text>
7          <Text>{siswa3}</Text>
8      </View>
9  )
```

Sedangkan jika ingin mengambil data pertama dan ketiga saja maka dapat dilakukan seperti pada potongan kode di bawah ini.

```
1  const siswa = ['budi', 'wati', 'iwan'];
2  const [siswa1, , siswa3] = siswa;
3
4  return (
5      <View>
6          <Text>Daftar Siswa</Text>
7          <Text>{siswa1}</Text>
8          <Text>{siswa3}</Text>
9      </View>
   )
```

Destructuring Obyek

Berikut ini adalah contoh bagaimana umumnya nilai-nilai pada obyek diambil. Berikut ini adalah cara yang paling mudah untuk membuat obyek. Selanjutnya untuk mengakses nilai-nilai property pada obyek dapat dilihat pada baris ke-8.

```
1  const person = {
2    firstName: "Iwan",
3    lastName: "Irawan",
4    age: 21,
5    gender: "man"
6  };
7
8  fullName = person.firstName + " " + person.lastName;
```

Pada Kode 14 dapat dilihat teknik destructuring obyek yang diimplementasikan pada aplikasi.

Kode 14. AppDestructuringObject.js.

AppDestructuringObject.js

```
1  import React from 'react';
2  import { Text, View } from 'react-native';
3  import { NavigationContainer } from '@react-navigation/native';
4
5  const AppDestructuringObject = () => {
6      const person = {
7          firstName: "Iwan",
8          lastName: "Irawan",
9          age: 21,
10         gender: "man"
11     };
12     const {firstName, lastName, age, gender} = person;
13
14     return (
15         <View>
16             <Text>Detail Siswa</Text>
17             <Text>Nama Lengkap: {firstName+" "+lastName}</Text>
18             <Text>Umur: {age}</Text>
19             <Text>Jenis Kelamin: {gender}</Text>
20         </View>
21     )
22 }
23
24 export default AppDestructuringObject;
```

Pada baris ke-12 dapat dilihat cara destructuring obyek. Setelah nilai-nilai obyek ditampung pada variable maka dapat ditampilkan pada layar aplikasi seperti yang dapat dilihat pada baris ke-17 sampai dengan ke-19.

10.Tenary Operator

Operator tenary adalah operator kondisional seperti if/else yang disederhanakan. Sintaks dari operator ini dapat dilihat seperti berikut ini.

```
condition ? <expression if true> : <expression if false>
```

Contohnya dapat dilihat pada Kode 15 di baris ke-6.

Kode 15. AppTernary.js

AppTernary.js

```
1  import React from 'react';
2  import { Text, View } from 'react-native';
3
4  const AppTernary = () => {
5      var nilai = 40;
6      var lulus = (nilai < 50) ? "Tidak Lulus" : "Lulus";
7
8      return (
9          <View>
10             <Text>Nilai Siswa</Text>
11             <Text>{nilai}: {lulus}</Text>
12         </View>
13     )
14 }
15
16 export default AppTernary;
```

Tugas

Lakukan tugas-tugas berikut:

1. Ikuti seluruh instruksi di atas.
2. Modifikasi Kode 15 agar nilai dimasukkan dalam komponen TextInput dan terdapat tombol yang akan mengeksekusi fungsi untuk menentukan apakah nilai yang dimasukkan adalah lulus atau tidak lulus. Status lulus dan tidak lulus ditampilkan ke layar dalam bentuk Alert atau Modal.

Referensi

1. https://www.w3schools.com/react/react_es6.asp