



BAB V

LIFECYCLE METHOD REACT

1. Lifecycle Component API's

Lifecycle component API's biasa dipakai untuk komponen react pada CLASS. Pada lifecycle class memiliki tiga buah fase utama yaitu, **Mounting**, **Updating**, dan **Unmounting**.

1.1. Mounting

Mounting merupakan Teknik untuk memasukan elemen kedalam bentuk DOM. Pada RCC memiliki 4 buah komponen yang perlu digunakan untuk melakukan Mounting:

- *constructor()*
Metode *constructor()* diinisialisasi pada awal pada saat komponen dimuat, dan ini adalah wadah untuk menyiapkan STATE awal dan beberapa nilai awal lainnya.
- *getDerivedStateFromProps()*
Metode ini biasanya dipanggil sebelum melakukan merender elemen di DOM. Metode ini baik digunakan untuk mengatur nilai value yang telah dikirimkan melalui props.
- *render()*
Metode *render()* merupakan metode yang benar-benar menampilkan HTML ke DOM atau yang dikenal dengan nama JSX.
- *componentDidMount()*
Metode *componentDidMount()* dipanggil setelah komponen dirender. Di sinilah metode ini menjalankan pernyataan yang mengharuskan komponen sudah harus ditempatkan pada DOM.

Berikut ini adalah contoh mounting untuk menangkap nilai dari sebuah inputan elemen JSX dengan menggunakan TextInput:

```
import { Dimensions, Image, SafeAreaView, StyleSheet, Text, TextInput, View, } from "react-native";
import React, { Component } from "react";

export class FormRCC extends Component {
  constructor(props){
    super(props);
    this.state={
      title:"IBI Kesatuan",
      subTitle:"Bogor Indonesia"
    }
  }
  render() {
    return (
      <SafeAreaView>
        <View>
          <Image source={require("../../../../../../assets/icons/icon-ibik.png")} />
          <View>
            <Text> {this.state.title} </Text>
            <Text> {this.state.subTitle} </Text>
          </View>
        </View>
      </SafeAreaView>
    )
  }
}
```

Inisialisasi key pada state lifecycle bernama title, dan subTitle

Cara menampilkan state lifecycle pada JSX

```

<View>
    <View>
        <Text>Change Logo</Text>
    </View>
    <View>
        <Text>Title</Text>
        <TextInput placeholder="Enter title here"
            defaultValue={this.state.title}
            onChangeText={(text)=>this.setState({title:text})} />
    </View>
    <View>
        <Text>Sub Title</Text>
        <TextInput placeholder="Enter sub title here"
            defaultValue={this.state.subTitle}
            onChangeText={(text)=>this.setState({subTitle:text})} />
    </View>
</SafeAreaView>
);
}

export default FormRCC;

```

Mengambil nilai value pada TextInput dengan menggunakan properties `onChangeText`, dan untuk mengupdate value pada state gunakan `this.setState({namakey:value})`

Dari script diatas maka output nya akan sebagai berikut:



Gambar 1.1. Tampilan output state lifecycle

Silakan anda ganti value pada isian form diatas untuk mengganti judul dan sub judul pada header aplikasi.

2. HOOK

Sedangkan untuk lifecycle pada React Function Component (RFC) menggunakan HOOK. Konsep dari HOOK ini jauh lebih simple dari React Class Component. Dengan menggunakan HOOK, state pada komponen dapat diakses dengan fitur react lainnya.

Dalam penggunaan HOOK ada tiga hal yang perlu diperhatikan, yaitu:

- Hook hanya dapat dipanggil di dalam komponen RFC.
- Hook hanya dapat dipanggil pada level teratas dari sebuah komponen.
- Hook tidak bisa berisi logika bersyarat

2.1. UseState

React `useState` Hook memungkinkan untuk negelola data pada state didalam komponen fungsi. State pada umumnya mengacu ke data atau properti yang perlu ditracking dalam aplikasi. Untuk menggunakan `useState` anda dapat menggunakan library dari react dengan cara mengimport

```
import { useState } from "react";
```

Berikut ini adalah contoh dari penggunaan HOOK namun dengan contoh kasus yang sama yaitu mengubah header aplikasi untuk title dan subtitle:

```
import { View, Text, StyleSheet, SafeAreaView, Image, TextInput } from "react-native";
import React, { useState } from "react";

const FormRFC = () => {
  const [title, setTitle] = useState("IBI Kesatuan");
  const [subTitle, setSubTitle] = useState("Bogor Indonesia");
  return (
    <SafeAreaView>
      <View>
        <Image source={require("../../../../../assets/icons/icon-ibik.png")} />
        <View>
          <Text>{title}</Text>
          <Text>{subTitle}</Text>
        </View>
      </View>
    </SafeAreaView>
  );
}

export default FormRFC;
```

Format penulisan `useState()`:
[getter, setter] = useState(value)

Inisialisasi title dan subtitle dengan menggunakan HOOK `useState`

Menampilkan nilai getter dari `useState`

Mengisi nilai baru pada setter title atau subtitle dengan properties `onChangeText(...)`



Dari script diatas maka outputnya tidak jauh beda dengan gambar 1.1.



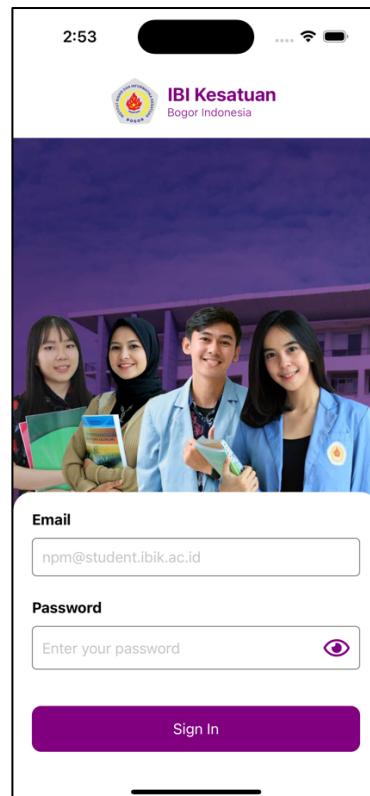
Gambar 2.1. Tampilan penggunaan HOOK

Selain

3. Operations

3.1. Selection

Pada contoh kali ini akan menggunakan tugas latihan praktikum sebelumnya, terdapat sebuah form Sign In yang berisikan Email dan Password. Berikut ini adalah contoh tampilan komponen Sign In dari materi sebelumnya:



Gambar 3.1.1. Tampilan komponen Sign In

Pada gambar diatas akan dibuat sebuah validasi dimana pembuatan validasi pada form tersebut akan menggunakan operation statement, contoh kali ini akan menggunakan operation statement model **IF ELSE**.



SignIn.js

```
import { View, Text, TextInput, StyleSheet, TouchableOpacity, Pressable} from "react-native";
import React, { useState } from "react";
import FontAwesome5 from "react-native-vector-icons/FontAwesome5";

const SignInBasic = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  return (
    <View style={styles.container}>
      <View style={styles.frm_row}>
        <Text style={styles.text_label}>Email </Text>
        <TextInput
          placeholder="npm@student.ibik.ac.id"
          keyboardType="email-address"
          autoCorrect={false}
          autoCapitalize="none"
          style={styles.text_input}
          defaultValue={email}
          onChangeText={(text) => setEmail(text)}
        />
      </View>

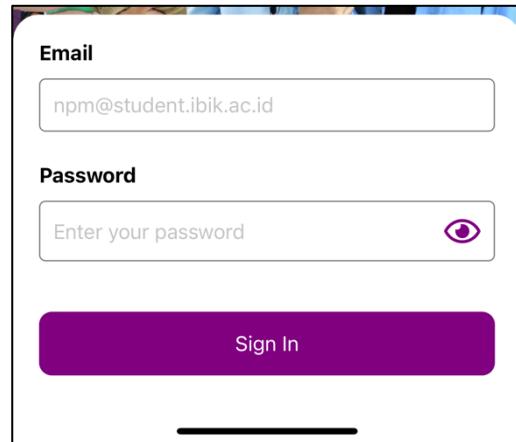
      <View style={styles.frm_row}>
        <Text style={styles.text_label}>Password</Text>
        <View
          style={{
            ...styles.text_input,
            flexDirection: "row",
            justifyContent: "space-between",
          }}
        >
          <TextInput
            secureTextEntry={true}
            placeholder="Enter your password"
            autoCorrect={false}
            style={{...styles.text_input, borderWidth:0, padding:0, width:"90%"}}
            defaultValue={password}
            onChangeText={(text)=>setPassword(text)}
          />
          <Pressable>
            <FontAwesome5 name={"eye"} size={25} color="purple" />
          </Pressable>
        </View>
      </View>
    </View>
  );
}
```



```
<TouchableOpacity style={styles.btn_signin} onPress={()=> handlerSignIn()} >
    <Text style={styles.btn_signin_text}>
        Sign In
    </Text>
</TouchableOpacity>
</View>
);
};

export default SignInBasic;

const styles = StyleSheet.create({
    container: {
        padding: 20, backgroundColor: "white"
    },
    frm_row: { marginBottom: 15 },
    text_label: {
        fontWeight: "bold",
        marginBottom: 10,
        fontSize: 16,
    },
    text_input: {
        borderWidth: 1,
        borderColor: "grey",
        borderRadius: 5,
        padding: 10,
        fontSize: 16,
        color: "purple",
    },
    btn_signin: {
        backgroundColor: "purple",
        paddingVertical: 15,
        paddingHorizontal: 10,
        borderRadius: 10,
        marginTop: 15
    },
    btn_signin_text:{ color: "white", textAlign: "center", fontSize: 16 }
});
```



Script diatas merupakan contoh bentuk skema dari RFC *Sign In* pada gambar 3.1.1, script tersebut akan dimodifikasi untuk membuat sebuah validasi, maka tahap pertama dari logika validasi ini ialah melakukan proses operation statement terhadap nilai masukan email dan password. Berikut ini ialah contoh tahapan untuk memberikan sebuah validasi sekaligus contoh penggunaan **IF ELSE**.

1. Membuat sebuah function untuk mengelola logika validasi email, function ES6 yang akan dibuat untuk mengelola logika tersebut ialah *handlerValidMail()*.



```
const SignInBasic = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const handlerValidMail = (value) => {
    //statement logic validasi email
  }
}
```

2. Didalam function tersebut memiliki parameter bernama value, dimana parameter tersebut yang akan menyimpan nilai balik dari TextInput email. Nilai balik tersebut akan dilakukan selection dengan IF ELSE. Kondisi pertama ialah, jika nilai parameter kosong maka mencetak "*This field is required*" pada JSX.

```
const [validEmail, setValidEmail] = useState("");
const handlerValidMail = (value) => {
  if(value){ //if value is not null
    setValidEmail("");
  }else{
    setValidEmail("This field is required");
  }
}
```

Lakukan perubahan script pada JSX *TextInput* email dengan mengubah nama tujuan function yang lama menjadi *handlerValidMail(...)*. Dan buatlah JSX untuk menampilkan pesan error dari validasi email tersebut.

```
<View style={styles.frm_row}>
  <Text style={styles.text_label}>Email </Text>
  <TextInput
    placeholder="npm@student.ibik.ac.id"
    keyboardType="email-address"
    autoCorrect={false}
    autoCapitalize="none"
    style={styles.text_input}
    defaultValue={email}
    onChangeText={(text) => handlerValidMail(text)}
  />
  <Text style={{ color:"red" }}>{validEmail}</Text>
</View>
```

Mengubah tujuan nama function

Menampilkan pesan error yang telah disetter pada function handlerValidMain(...) dgn memanggil getter dari validEmail

Maka output sementara dari seleksi kondisi pertama akan sebagai berikut:

Email

hpm@student.ibik.ac.id

This field is required

3. Kondisi statement kedua kali ini akan mengecek apakah inputan email memiliki format yang benar atau tidak. Untuk mengecek format expression pada JAVASCRIPT dapat menggunakan Teknik REGEX.

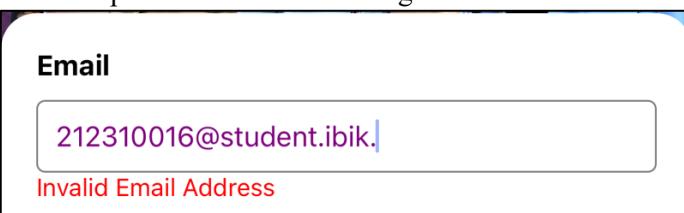
Berdasarkan format expression REGEX yang dilihat dari https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions
Maka dapat expression kan untuk email menjadi:

Email	Expression Regex
name@domain.com	/^(([^\<()\\]\\.,;:\\s@"]+(\.([^\<()\\]\\.,;:\\s@"]+)* (".+"))@((\\[[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}]) (([a-zA-Z\\-0-9]+\\.){2,}))\$/

4. Berdasarkan konsep data regex diatas maka dapat kita buat selection untuk mengecek apakah nilai inputan email sudah benar atau tidak:

```
const [validEmail, setValidEmail] = useState("");
const handlerValidMail = (value)=>{
  if(value){ //if value is not null
    let regEmail = /^(([^\<()\\]\\.,;:\\s@"]+(\.([^\<()\\]\\.,;:\\s@"]+)*|(".+"))@((\\[[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}])|(([a-zA-Z\\-0-9]+\\.){2,}))$/
    if (!regEmail.test(value)) {
      setValidEmail("Invalid Email Address");
    } else {
      setValidEmail("");
    }
  }else{
    setValidEmail("This field is required");
  }
}
```

Maka output sementara ialah sebagai berikut:



The screenshot shows a simple form with a single input field labeled "Email". The input field contains the text "212310016@student.ibik.". Below the input field, the error message "Invalid Email Address" is displayed in red text.

5. Contoh kasus kedua ialah penggunaan selection dalam bentuk SWITCH CASE. Penggunaan selection bentuk kedua ini akan diimplementasikan untuk membuka dan tutup value pada isian password. Logika kali ini akan disimpan dengan sebuah function bernama *handlerOpenPassword()*.

Logika untuk membuka dan menutup isian password ini akan mengambil trigger ketika mengklik icon EYE maka akan membuka isian password begitu sebaliknya.

Tambahkanlah function dibawah ini pada script RFC SignIn sebelumnya:

```
const [isOpenPass, setOpenPass] = useState(true);
const handlerOpenPassword = () =>{
  switch (!isOpenPass) {
    case true:
      setOpenPass(true);
      break;
    default:
      setOpenPass(false);
      break;
  }
}
```

Pada script diatas selain menambahkan function baru bernama `handlerOpenPassword`, ditambahkan juga sebuah variable `hook` untuk menyimpan nilai Boolean terhadap aksi buka dan tutup password dengan nama `isOpenPass` sebagai current variable (getternya) dan `setOpenPass` sebagai bentuk pengisian nilai value baru (setternya).

Selanjutnya diperlukan untuk memperbarui elemen JSX bagian password agar dapat memperlihatkan isian dari `TextInput` dan memberikan triger terhadap icon EYE agar dapat mengakses function `handlerOpenPassword()`.

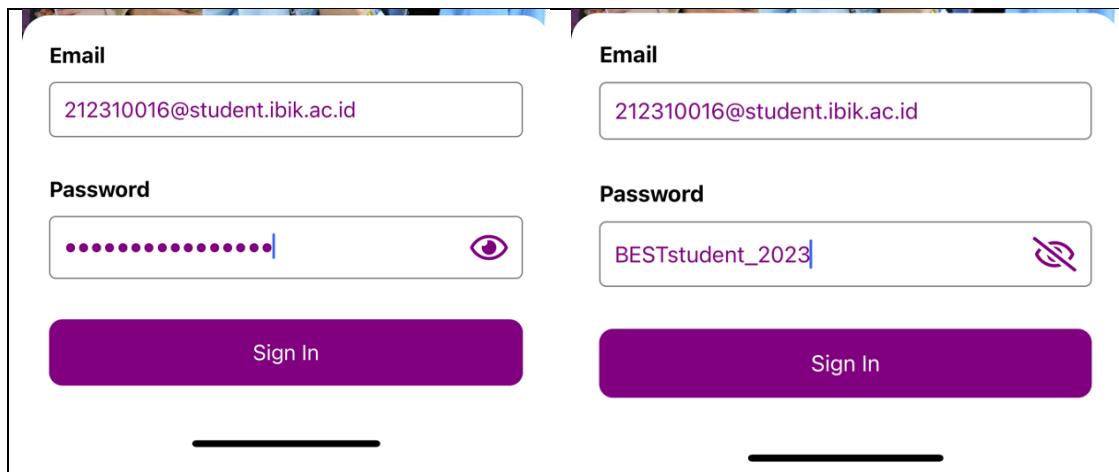
```

<TextInput
  secureTextEntry={isOpenPass}
  placeholder="Enter your password"
  autoCorrect={false}
  style={{ ...styles.text_input, borderWidth:0, padding:0, width:"90%" }}
  defaultValue={password}
  onChangeText={({text})=>setPassword(text)}
/>
<Pressable onPress={()=>handlerOpenPassword()}>
  <FontAwesome5 name={(isOpenPass) ? "eye":"eye-slash"} size={25} color="purple" />
</Pressable>

```

Bentuk SHORT IF pada JSX

Dari aksi script diatas maka output sementara seperti berikut ini:



Gambar 3.1.5. Tampilan validasi password

3.2. Repetition

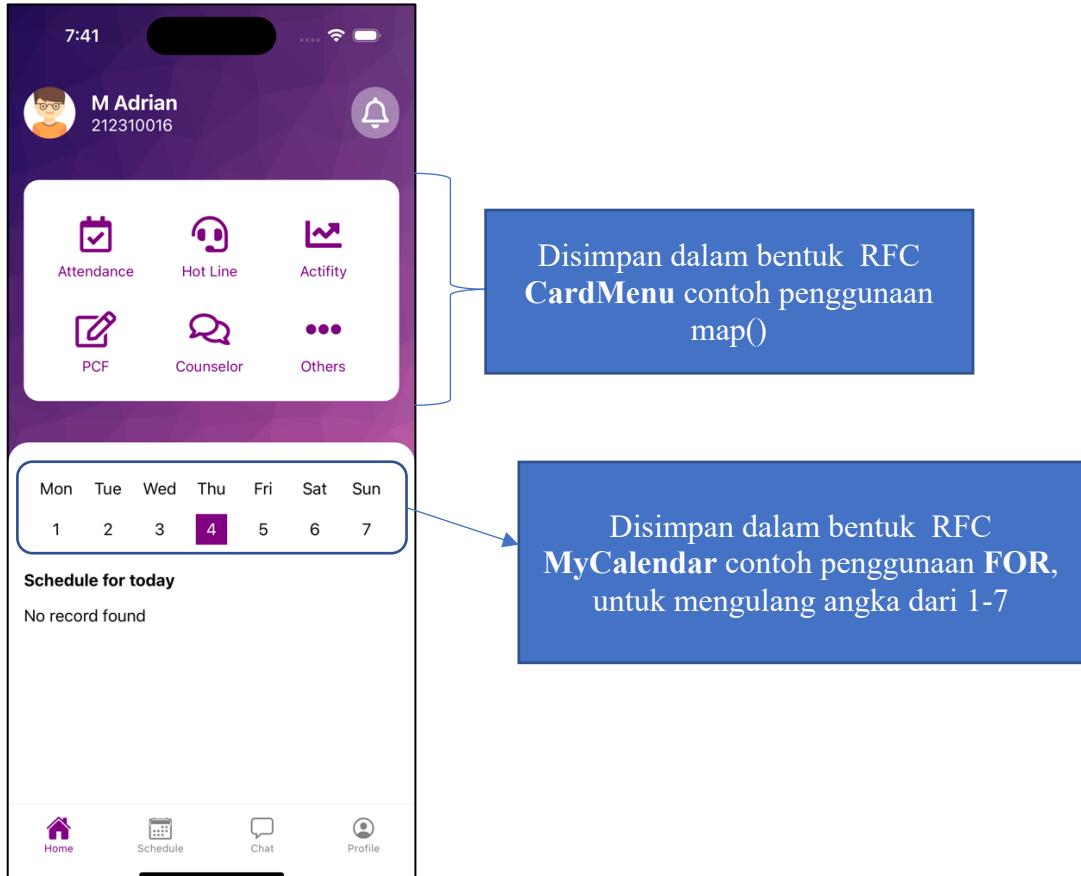
Untuk tipe-tipe dari repetition sama halnya dengan beberapa Bahasa pemograman lainnya, yaitu memiliki bentuk :

- Do { statement }while(condition)
- While(condition){ statement }
- For(condition){ statement }, dan

Dari ketiga bentuk repetition diatas, sama halnya dengan selection. Bentuk repetition tersebut tidak bisa ditulis kedalam render JSX.

Untuk mengeksekusi perintak logika repetition disarankan membentuk function tersendiri dan mereturn nilai balik dalam bentuk JSX.

Namun pada JAVASCRIPT ada 1 bentuk repetition yang dapat digunakan didalam JSX ataupun diluar JSX, yaitu menggunakan `map()`. Berikut ini adalah contoh penggunaan repetition dalam bentuk FOR dan map:



Gambar 3.2. Tampilan penggunaan Repetition

CardMenu.js

```
import { View, Text, StyleSheet } from "react-native";
import React from "react";
import FontAwesome5 from "react-native-vector-icons/FontAwesome5";

const CardMenu = () => {
  const menuList = [
    { id: 1, name: "Attendance", icon: "calendar-check" },
    { id: 2, name: "Hot Line", icon: "headset" },
    { id: 3, name: "Actifity", icon: "chart-line" },
    { id: 4, name: "PCF", icon: "edit" },
    { id: 5, name: "Counselor", icon: "comments" },
    { id: 6, name: "Others", icon: "ellipsis-h" },
  ];

  return (
    <View>
      <View style={{ flexDirection:"row", flexWrap:"wrap" }}>
        {menuList.map((v, index) => (
          <View style={{ margin: 10 }}>
            <Text>{v.name}</Text>
            <Image icon={v.icon} style={{ width: 24, height: 24 }} />
          </View>
        ))}
      </View>
    </View>
  );
}
```



```
        <MenuItem item={v} key={index} />
    ))
</View>
</View>
);
};

export default CardMenu;

const MenuItem = ({ item }) => {
    return (
        <View style={styles.card_item}>
            <View style={styles.card_item}>
                <FontAwesome5 name={item.icon} size={35} color={"purple"} />
            </View>
            <Text style={styles.card_text}>{item.name}</Text>
        </View>
    );
};

const styles = StyleSheet.create({
    card_item:{
        width:100,marginHorizontal:5, marginVertical:10, flexDirection:"column",
        justifyContent:"center", alignItems:"center"
    },
    card_icon:{marginBottom:10},
    card_text:{ color:"purple", fontSize:14 }
});
```

MyCalendar.js

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

const CalendarFirstWeek = () => {
    const FirstWeek = () => {
        const days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"];
        var day = [];
        for (let index = 0; index < 7; index++) {
            day.push(
                <View key={index} style={styles.call_box}>
                    <Text style={{ ...styles.call_text, marginBottom:10 }}>
                        {days[index]}</Text>
                    <Text>{index + 1}</Text>
                </View>
            );
        }
    }
}
```



```
        return day;
    };

    return (
        <View style={styles.callendar_container}>
            <FirstWeek />
        </View>
    );
};

export default CalendarFirstWeek;

const styles = StyleSheet.create({
    callendar_container: {
        flexDirection: "row",
        alignItems: "center",
        justifyContent: "center",
        marginVertical: 15,
    },
    call_box: {
        width: 50,
        flexDirection: "column",
        alignItems: "center",
    },
    call_text:{
        fontSize:16,
        paddingVertical:5,
    },
    call_curr:{ 
        padding:10,
        backgroundColor:"purple",
        color:"white"
    }
});
```

3.3. Operasi Aritmatika

Bentuk dari operasi aritmatika pada JAVASCRIPT sebenarnya tidaklah terlalu berbeda dengan Bahasa pemrograman umum lainnya, berikut ini adalah contoh operasi aritmatika yang dapat digunakan pada JAVASCRIPT:

Operator	Keterangan
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
>	Lebih Besar
<	Lebih Kecil
>=	Lebih besar sama dengan
<=	Lebih kecil sama dengan

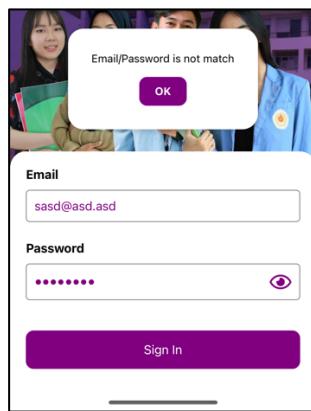


====	Sama dengan / equals
&	AND
	OR
++	Increments
--	Decrements
%	Modulus
Length	Mendapatkan nilai total data dari variable
Object.key()	Mendapatkan nama-nama key pada object
Object.value()	Mendapatkan nilai value pada object

4. Latihan Praktikum

1. Buatlah package baru bernama praktikum-5.
2. Melanjutkan codingan pada contoh **Gambar 3.1.5**, buatlah sebuah validasi untuk password dengan kondisi sebagai berikut:
 - a. Jika jumlah password kurang dari 3 maka menampilkan pesan "*Type minimum 3 character*".
 - b. Jika kondisi value password memiliki bentuk isian berupa: huruf, angka, dan sysmbol maka kondisi password dinyatakan lulus, namun jika tidak memenuhi kondisi ini buatlah pesan kesalahan "*Value must contain alphabet, number and symbol*"
3. Jika soal nomor 2 sudah anda kerjakan buat lah sebuah event handler untuk mengeksekusi kedua buah isian tersebut. Jika mengklik tombol SIGN IN maka akan mengeksekusi logika untuk:
 - a. Jika Email yang dimasukan berisi 212310016@student.ibik.ac.id
 - b. Dan password yang dimasukan berisi BESTstudent_2023

Maka dinyatakan memenuhi kondisi transaksi, dan dapat di redirect ke halaman berikutnya. Namun jika tidak memenuhi kondisi diatas maka menampilkan pesan kesalahan dalam bentuk Pop Up "*Email/Password is not match*".



Pengumpulan tugas Latihan praktikum dikumpulkan kedalam GITHUB masing-masing mahasiswa berdasarkan repository yang telah dibuat PPB-TI-21-[PA/KA]-NPM. File source code disimpan sesuai nama project-praktikum dan masukan kedalam repositori tersebut. Buatkanlah file dokumen dalam bentuk file pdf yang berisi Screen Capture dari hasil program yang telah dikerjakan. Simpan dalam file PDF tersebut kedalam project tersebut.

Tambahkan Collaborator management access pada repository anda kepada:



Matakuliah/Code
Dosen
Kelas

: Lab. Pemrograman Perangkat Bergerak / TIFA3P3
: Irvan Rizky Ariansyah / Thesya Marcella
: TI-21-PA

@FebryFairuz dan (@IrvanRizkyAriansyah atau @thesyamarcella)

Disusun Oleh	Disetujui Oleh
<i>Thesya Marcella Penyusun I</i>	<i>Irvan Rizky Ariansyah Penyusun II</i>