

# React Native

## React Components

---



# React Component

- ☐ Folder Structure Introduction
- ☐ HTML dan JSX
- ☐ Rendering Dynamic JSX
- ☐ State and Props

# Objektif sesi

- Memahami Folder Struktur pada React Native Project
- Memahami Penggunaan JSX
- Memahami State dan Props

# Folder Structure

Struktur folder proyek pada Expo React Native mengikuti beberapa konvensi tertentu. Berikut adalah gambaran umum dari susunan folder struktur proyek Expo React Native menggunakan referensi dari [dokumentasi Expo](#):

```
my-expo-project/  
|-- .expo/  
|-- assets/  
|-- node_modules/  
|-- .gitignore  
|-- .watchmanconfig  
|-- App.js  
|-- app.json  
|-- babel.config.js  
|-- package.json  
|-- README.md  
|-- metro.config.js
```



# HTML dan JSX

React Native tidak menggunakan HTML untuk membangun antarmuka pengguna seperti pada pengembangan web tradisional.

Sebaliknya, React Native menggunakan JSX (JavaScript XML) untuk membuat antarmuka pengguna pada aplikasi seluler. JSX mirip dengan HTML, tetapi sebenarnya merupakan ekstensi sintaks JavaScript yang memungkinkan penulisan kode yang lebih dekat dengan HTML.

```
// Import React dan komponen dari React Native
import React from 'react';
import { View, Text, StyleSheet } from 'react-native';

// Komponen utama
const MyApp = () => {
  return (
    // Komponen View sebagai wadah
    <View style={styles.container}>
      {/* Komponen Text untuk menampilkan teks */}
      <Text style={styles.text}>Hello, React Native!</Text>
    </View>
  );
};

// Objek StyleSheet untuk mendefinisikan gaya komponen
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#f0f0f0',
  },
  text: {
    fontSize: 24,
    fontWeight: 'bold',
    color: '#333',
  },
});

// Ekspor komponen agar dapat digunakan di tempat lain
export default MyApp;
```

# HTML dan JSX

```
// Import React dan komponen dari React Native
import React from 'react';
import { View, Text, StyleSheet } from 'react-native';

// Komponen utama
const MyApp = () => {
  return (
    // Komponen View sebagai wadah
    <View style={styles.container}>
      {/* Komponen Text untuk menampilkan teks */}
      <Text style={styles.text}>Hello, React Native!</Text>
    </View>
  );
};

// Objek StyleSheet untuk mendefinisikan gaya komponen
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#f0f0f0',
  },
  text: {
    fontSize: 24,
    fontWeight: 'bold',
    color: '#333',
  },
});

// Ekspor komponen agar dapat digunakan di tempat lain
export default MyApp;
```

- **View** adalah komponen utama yang berfungsi sebagai wadah untuk elemen lainnya. Flexbox digunakan untuk menata elemen di dalamnya.
- **Text** adalah komponen yang digunakan untuk menampilkan teks pada antarmuka pengguna.
- **StyleSheet.create** digunakan untuk mendefinisikan gaya komponen dengan objek StyleSheet. Ini memungkinkan penerapan gaya yang efisien dan mudah dipelihara.
- **Export** komponen memungkinkan komponen ini digunakan di file atau komponen lain.


# State

State adalah objek yang dimiliki oleh komponen dan digunakan untuk menyimpan data lokal di dalam komponen tersebut.

State dapat diubah oleh komponen yang memiliki state, dan ketika state berubah, komponen akan dire-render untuk memperbarui tampilan.

State bersifat privat dan hanya dapat diakses oleh komponen yang memiliki state tersebut.

jsx

 Copy code

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

const CounterApp = () => {
  // Mendefinisikan state 'count' dan fungsi untuk mengubahnya ('setCount')
  const [count, setCount] = useState(0);

  const increment = () => {
    // Mengubah nilai state 'count' ketika tombol ditekan
    setCount(count + 1);
  };

  return (
    <View>
      <Text>Counter: {count}</Text>
      <Button title="Increment" onPress={increment} />
    </View>
  );
};

export default CounterApp;
```

# Props

jsx

Copy code

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

// Komponen anak (ChildComponent) dengan props
const ChildComponent = (props) => {
  return (
    <View>
      <Text>{props.message}</Text>
    </View>
  );
};

// Komponen induk (ParentComponent) yang melewati prop ke ChildComponent
const ParentComponent = () => {
  return (
    <View>
      /* Menyertakan komponen ChildComponent dan meneruskan prop 'message' */
      <ChildComponent message="Hello from ParentComponent!" />
    </View>
  );
};

export default ParentComponent;
```

Props (singkatan dari properties) adalah nilai yang diteruskan dari komponen induk ke komponen anak.

Props bersifat read-only, artinya komponen anak tidak dapat mengubah nilai prop yang diterimanya dari komponen induk.

Props digunakan untuk mengirim data dari satu komponen ke komponen lain dan membantu dalam membuat komponen yang dapat digunakan kembali.



# Rendering Dynamic using State

Penggunaan komponen **TextInput** untuk memungkinkan pengguna memasukkan nilai dinamis.

Nilai dinamis tersebut disimpan dalam **state dynamicValue** menggunakan **useState**.

Komponen **Text** digunakan untuk **merender nilai dinamis** tersebut.

Penggunaan nilai dinamis dalam JSX juga diperlihatkan, di mana kita menggabungkan nilai dinamis dalam string dan merendernya sesuai kebutuhan.

Sebuah tombol "Clear" ditampilkan ketika nilai dinamis tidak kosong, dan pengguna dapat menggunakan tombol ini untuk mengosongkan nilai dinamis.

```
import React, { useState } from 'react';
import { View, Text, TextInput, Button } from 'react-native';

const DynamicValueApp = () => {
  // State untuk menyimpan nilai dinamis
  const [dynamicValue, setDynamicValue] = useState('');

  // Fungsi untuk mengubah nilai dinamis
  const updateDynamicValue = (text) => {
    setDynamicValue(text);
  };

  return (
    <View>
      {/* Komponen TextInput untuk memasukkan nilai dinamis */}
      <TextInput
        style={{ height: 40, borderColor: 'gray', borderWidth: 1, margin: 10 }}
        placeholder="Enter dynamic value"
        onChangeText={updateDynamicValue}
        value={dynamicValue}
      />

      {/* Komponen Text untuk merender nilai dinamis */}
      <Text style={{ fontSize: 18, margin: 10 }}>Dynamic Value: {dynamicValue}</Text>

      {/* Contoh penggunaan nilai dinamis dalam JSX */}
      {dynamicValue !== '' && (
        <View>
          <Text style={{ fontSize: 16, margin: 10 }}>
            {'You entered: ${dynamicValue}'}
          </Text>
          <Button title="Clear" onPress={() => setDynamicValue('')} />
        </View>
      )}
    </View>
  );
};

export default DynamicValueApp;
```



# Thank you