



**BINAR**  
ACADEMY

## **Redux Middleware, REST API dengan Redux, dan Persistence Storage**

**Gold** - Chapter 4 - Topic 3

---

**Selamat datang di Chapter 4 Topic 3 online course  
React Native dari Binar Academy!**





Alohaa kawan! 🙌

Akhirnya kita sampai di topic terakhir chapter 4 nih! Yang masih semangat mana suaranyaaa???

Nah, di topic ketiga ini, kamu akan belajar tentang **Redux Middleware**, **REST API dengan Redux**, dan **Persistence Storage**.

Nggak pake lama, cus langsung aja!



**Detailnya, kita bakal bahas hal-hal berikut ini:**

- Mengenal Redux Thunk Middleware
- CRUD with Redux
- Redux Persist





Guys, masih inget dengan step by step dari Redux?

Nah, di topic ini ada step tambahan di antara step by step Redux yang udah kamu pelajari sebelumnya, namanya **Redux Middleware**

Yuk langsung kita kepoin~



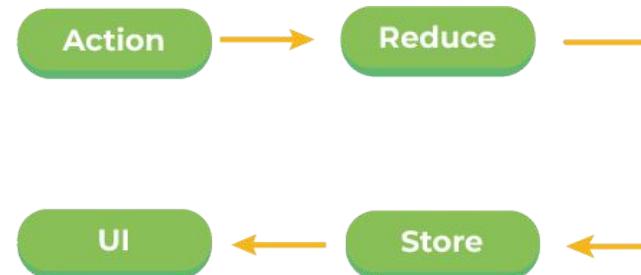


### Throwback sedikit ke step by step dalam Redux dulu ya!

Pada dasarnya Redux hanyalah proses operasi menyimpan dan mengupdate data pada sebuah store.

Jadi step-step dalam Redux itu simplenya :

**Trigger action -> update reducer -> save data di store ->  
update data di UI**





## Ini dia step tambahannya, Redux Middleware

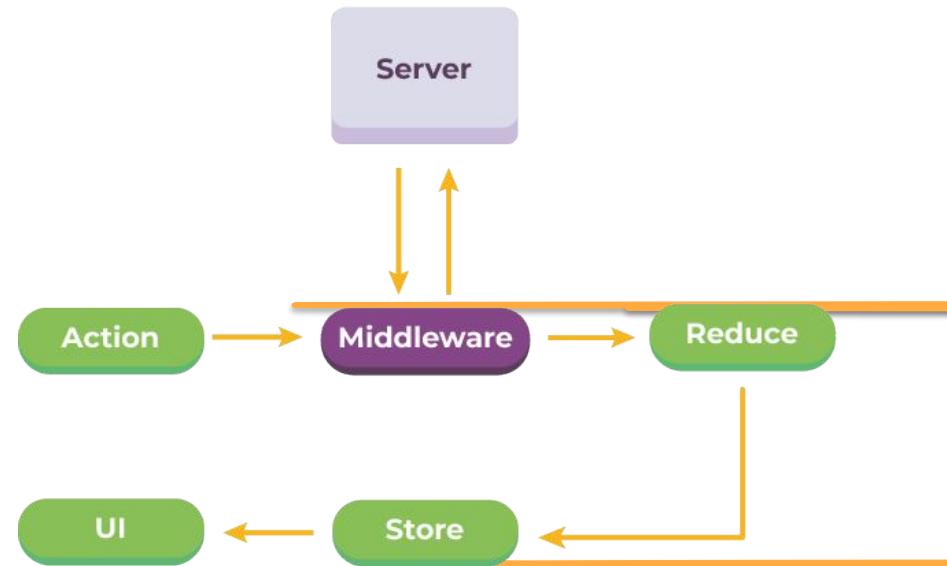
Redux Middleware adalah proses di mana kamu menambahkan satu proses lagi setelah step trigger action dan sebelum update reducer dalam Redux.





## Kenapa perlu proses tambahan?

Karena supaya kamu bisa melakukan HTTP Request atau mendapatkan data dari server, sehingga data dari server tersebut dapat disimpan di dalam store.





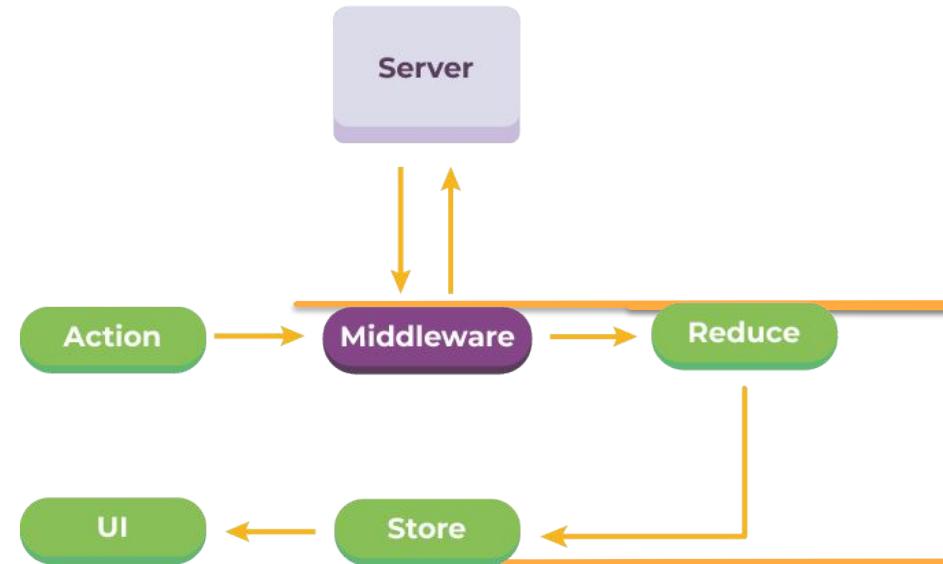
## Proses step by step Redux pun akan berubah

Jadi proses awal sebelum ada Redux Middleware kan begini:

**Trigger action -> update Reducer -> save data di Store -> update data di UI**

Dengan menggunakan Redux middleware jadi begini:

**Trigger action -> HTTP Request -> update Reducer -> save data di Store -> update data di UI**





### Jenis-jenis Redux middleware

Redux Middleware ini ada banyak jenis, guys, tapi ada 3 yang umum digunakan yaitu:

1. Redux Thunk
2. Redux Saga
3. Redux Promise Middleware

Nah, pada topic ini kita akan pilih dan menggunakan Redux-Thunk sebagai topic bahasan dan studi kasus.

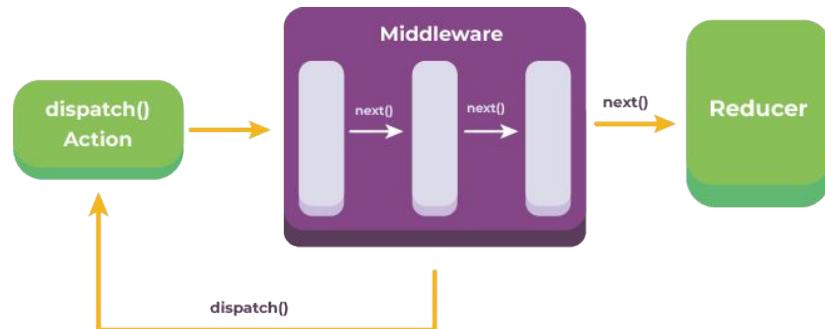




### Apa itu Redux-Thunk?

Redux Thunk merupakan sebuah middleware yang digunakan agar kamu bisa bikin Action yang mengembalikan sebuah function sebagai ganti objek.

Penggunaan Redux Thunk memungkinkanmu untuk menggunakan fungsi Async dalam Redux.

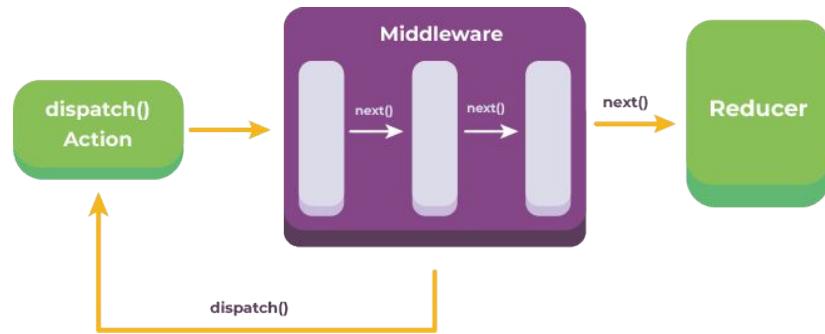




### Apa fungsi Redux-Thunk?

Redux-Thunk sangat berguna kalau kamu mau melakukan pemanggilan API atau memakai third-party library seperti Firebase dll.

Kita akan membahas lebih lanjut tentang Redux-Thunk pada saat pembuatan aplikasi ya.





## Kayak apa sih contoh Redux-Thunk?

Nah, di samping adalah contoh aplikasi Redux-Thunk pada code untuk melakukan pemanggilan API, gengs.

Okee, setelah memahami teori dari Redux dan Redux-Thunk, belum afdol rasanya kalau kamu nggak langsung bikin aplikasinya.

Bikin yuk!

```
● ● ●  
//Define your action creators that will be responsible for asynchroneuse operations  
export const getPeople = () => {  
  //IN order to use await your callback must be asynchronous using async keyword.  
  return async dispatch => {  
    //Then perform your asynchronous operations.  
    try {  
      //Have it first fetch data from our starwars url.  
      const starWarsPromise = await fetch('https://swapi.co/api/people');  
      dispatch(fetchData(true));  
  
      const people = await starWarsPromise.json();  
  
      dispatch(fetchDataFulfilled(people.results))  
    } catch(error) {  
      dispatch(fetchDataRejected(error))  
    }  
  }  
}
```



Guys, masih inget CRUD kan? Create, Read, Update, Delete

Nah kali ini kita akan melakukan CRUD dengan menggunakan Redux. Gimana caranya?

Yuk simak ya!





**Sekali dayung, dua tiga pulau terlewati~**

**Nah, belajar CRUD menggunakan Redux nya dalam bentuk pembuatan aplikasi ya, guys! Biar makin jago!**



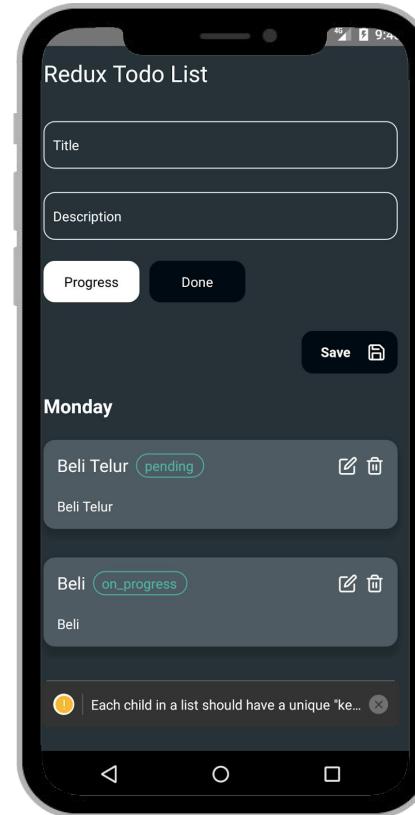


## Mari kita bikin~

Nah, di sesi ini kamu belajar untuk membuat sebuah aplikasi TODO List menggunakan Redux, Redux Thunk dan React Native.

Fitur aplikasi yang akan kamu bikin adalah mengambil **list TODO** dan **HTTP Request pada Redux-thunk**.

Cuss perhatikan langkah-langkahnya ya!

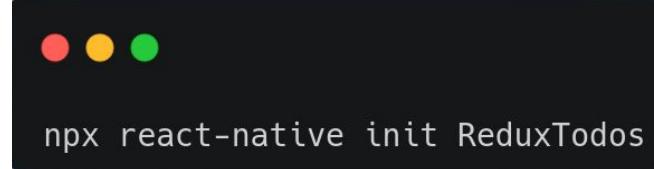




## 1. Setup Project React Native

Kita langsung mulai aja ya! Langkah pertama yang harus kamu lakukan adalah membuat sebuah project react native baru dengan menggunakan perintah:

**npx react-native init <NAMA-PROJECT-KAMU>**



```
npx react-native init ReduxTodos
```



## 2. Install Redux dan Redux Thunk

Proses kedua adalah install library-library yang dibutuhkan untuk integrasi Redux dengan react native. Di sini ada dua library yang wajib kamu install yaitu **React-Redux** dan **Redux**.



```
npm install react-redux redux redux-thunk
```



Karena kamu memakai Redux-Thunk sebagai middleware,  
maka kamu juga wajib menginstall **Redux-Thunk**.

Pada command terminal ketikan perintah :

**npm install react-Redux Redux-thunk**



```
npm install react-redux redux redux-thunk
```



### 3. Jalankan React Native

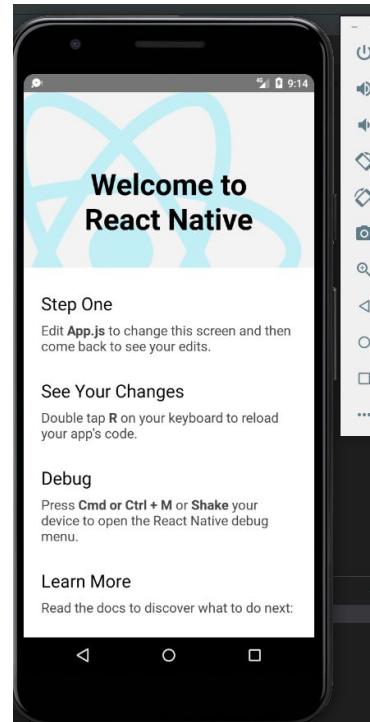
Setelah proses instalasi library Redux berhasil, pastikan project react native kamu dapat dijalankan.

untuk menjalankan aplikasi pada device android maka dilakukan dengan menggunakan perintah:

**npm run android**

Kalau di devices iOS pakai perintah::

**npm run ios**

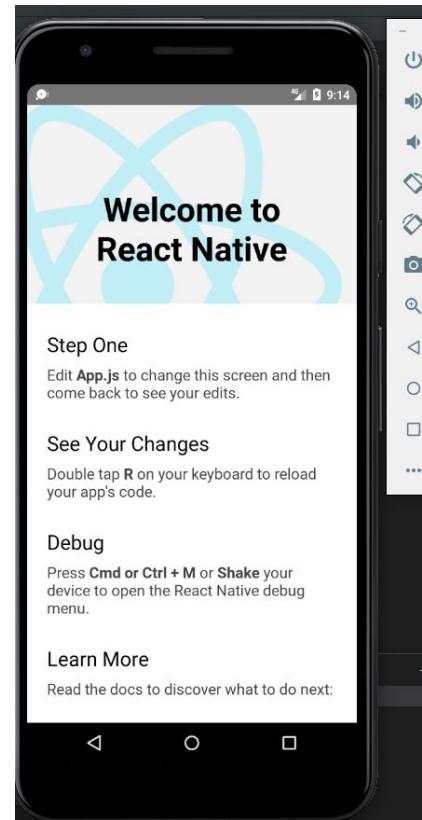




## 4. Setup Redux dan Redux-Thunk

Kamu sudah berhasil install package yang kamu perlukan untuk menggunakan Redux dan Redux-Thunk.

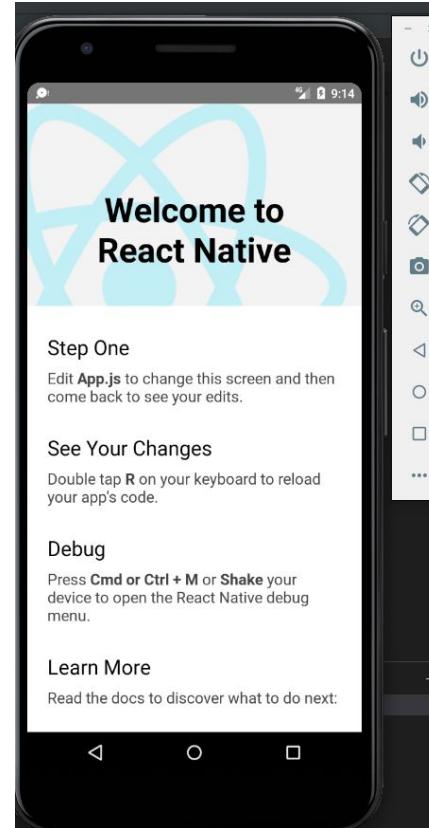
Langkah selanjutnya adalah yang kamu perlu menyusun folder tempat menyimpan dan mengatur untuk elemen-elemen pada Redux yaitu Action, Reducer dan Store.





## React Native

Di dalam folder **src**, buatlah tiga buah folder, yaitu folder **actions, reducers, store, types, dan container.**





## 5. Folder Types

Folder ini akan kamu pakai untuk menyimpan ragam types Action yang kamu gunakan. Tujuan folder ini adalah untuk mempermudah kamu dalam konfigurasi Action dan Reducer-nya.

Di dalam folder types ini buatlah sebuah file dengan nama index.js



```
export const FETCH_TODO_SUCCESS = '@FETCH_TODO_SUCCESS';
```



## 6. Folder Actions

Folder ini berguna untuk membuat berbagai macam logic yang diperlukan pada **Action**.

Di dalam folder **Actions** ini buatlah sebuah file dengan nama **index.js**, kemudian isi file **index.js** tersebut dengan Action **saveTodos** seperti pada syntax gambar di samping.

```
● ● ●  
import {  
  FETCH_TODO_SUCCESS  
} from '../Types';  
  
export const saveTodos = (data) => ({  
  type: FETCH_TODO_SUCCESS,  
  payload: data  
});
```



## 7. Folder Reducers

Folder ini kita gunakan untuk menentukan state apa saja yang harus disimpan pada Redux.

Di dalam folder **Reducers** ini buatlah sebuah file dengan nama **index.js**, kemudian isikan file **index.js** tersebut dengan dengan syntax kode di samping.

```
● ● ●

import { FETCH_TODO_SUCCESS } from '../Types';
const initialState = {
  todos: ''
};

const Reducer = (state = initialState, action) => {
  switch (action.type) {
    case FETCH_TODO_SUCCESS:
      return {
        ...state,
        todos: action.payload,
      };
    default:
      return state;
  }
};
export default Reducer;
```



Perhatikan deh pada variabel **initialState**, di sana kamu bikin sebuah object dengan property **todos** di dalamnya.

Nah, nantinya property **todos** ini lah tempat menyimpan semua data yang berkaitan dengan **todos**.

Property todos pada folder Reducer ini nantinya akan disimpan pada store di Redux.



```
import { FETCH_TODO_SUCCESS } from '../Types';
const initialState = {
  todos: ''
};

const Reducer = (state = initialState, action) => {
  switch (action.type) {
    case FETCH_TODO_SUCCESS:
      return {
        ...state,
        todos: action.payload,
      };
    default:
      return state;
  }
};
export default Reducer;
```



## 8. Folder Store

Folder Store ini akan kamu gunakan untuk menyimpan konfigurasi utama dari Redux, di mana kamu juga akan mendefinisikan middleware Redux-Thunk yang akan digunakan pada aplikasi ini.

```
import {applyMiddleware, combineReducers, createStore } from 'redux';
import ReduxThunk from 'redux-thunk';

import TodosReducer from '../Reducers';

const Reducers = {
    appData: TodosReducer
}

export const Store = createStore(combineReducers(Reducers), applyMiddleware(ReduxThunk))
```



Di dalam folder **Store**, buatlah sebuah file dengan nama **index.js**

Dapat dilihat bahwa kamu mendefinisikan middleware Redux-thunk dengan menggunakan command **applyMiddleware(thunk)**

```
import {applyMiddleware, combineReducers, createStore } from 'redux';
import ReduxThunk from 'redux-thunk';

import TodosReducer from '../Reducers';

const Reducers = {
    appData: TodosReducer
}

export const Store = createStore(combineReducers(Reducers), applyMiddleware(ReduxThunk))
```



## 9. Folder Container

Folder ini akan digunakan untuk menyimpan code yang akan jadi page atau screen-mu.

Folder ini berisi kode yang berkaitan dengan component-component react native yang akan ditampilkan pada screen aplikasi.

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

export default class Todos extends React.Component {

  render() {
    return (
      <Text>Hello</Text>
    )
  }
}
```



Sekarang kita bikin satu screen dulu dengan code awal seperti gambar di samping. Nanti kita akan melakukan banyak modifikasi lebih lanjut untuk file ini .

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

export default class Todos extends React.Component {

  render() {
    return (
      <Text>Hello</Text>
    )
  }
}
```



## 10. Update File App.js

Nah, pada file **App.js** yang terletak pada root project, kamu akan menambahkan component **<Provider>** yang disediakan oleh **React-Redux**.

Ini berfungsi sebagai parent/wrapper yang akan bertanggung jawab terhadap Redux yang kamu pakai di aplikasi, sehingga setiap state yang disimpan di Redux dapat diakses oleh semua component di bawahnya.

```
● ● ●

import React from 'react';
import {
  SafeAreaView,
  ScrollView,
  useColorScheme,
} from 'react-native';
import {
  Colors,
} from 'react-native/Libraries/NewAppScreen';
import { Provider } from 'react-redux';
import { Store } from './Store';
import Todos from './Containers/Todos';

const App = () => {
  const isDarkMode = useColorScheme() === 'dark';

  const backgroundStyle = {
    backgroundColor: isDarkMode ? Colors.darker : Colors.lighter,
  };

  return (
    <Provider store={Store}>
      <SafeAreaView style={backgroundStyle}>
        <ScrollView
          contentInsetAdjustmentBehavior="automatic"
          style={backgroundStyle}>
          <Todos />
        </ScrollView>
      </SafeAreaView>
    </Provider>
  );
};

export default App;
```

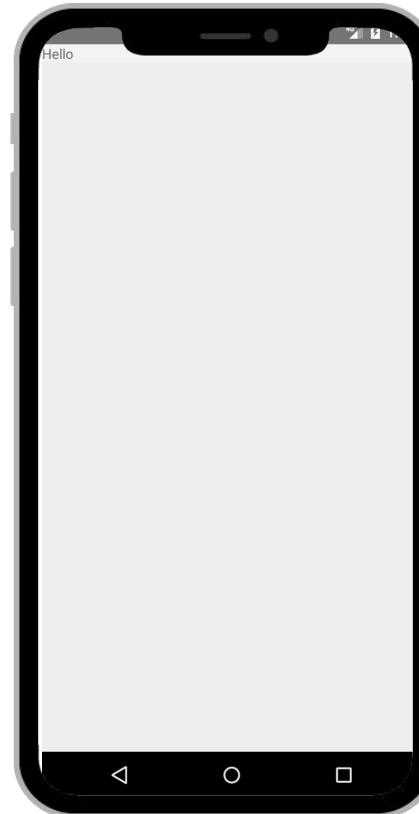


## 11. Running ulang/reload project react native

Setelah selesai mengedit file **App.js**, running ulang atau build ulang aplikasi kamu menggunakan command:

**npm run android**

Aplikasi kamu akan menampilkan tampilan seperti gambar di samping.

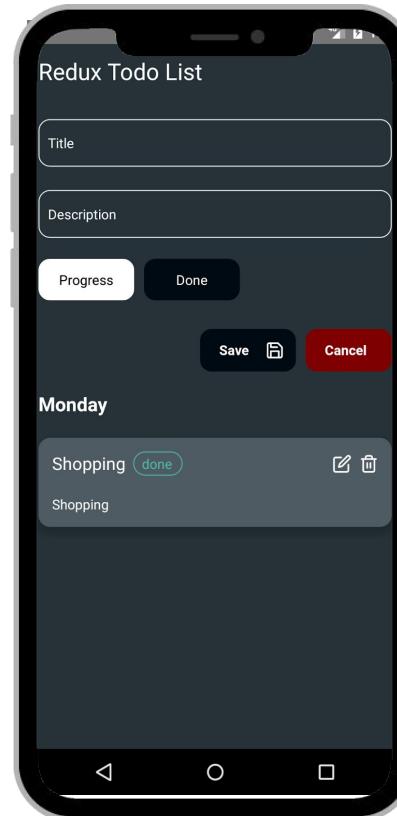




## 12. Setup UI

Untuk UI-nya kamu akan membuatnya tetap simple dan sederhana, kemudian untuk menambahkan icon kamu akan menggunakan package react-native-vector-icon.

Cara menginstall react-native-vector icons dapat dilihat pada [link ini](#) ya!





Pada file Containers/Todos/index.js tuliskan syntax kode untuk membuat tampilan seperti gambar di samping. Syntax kode tersebut dapat kamu liat pada [link ini](#).



```
import React from 'react';
import {
  View,
  TouchableOpacity,
  Text,
  TextInput
} from 'react-native';
import Icon from 'react-native-vector-icons/Feather';

import styles from './styles';

export default class Todos extends React.Component {

  constructor(props) {
    super(props);
    this.state = {
      title: '',
      description: '',
      selectedStatus: 'ON_PROGRESS',
    }
  }

  render() {
    return (
      <View style={styles.root}>
        <View style={styles.headingContainer}>
          <Text style={styles.heading}>Redux Todo List</Text>
        </View>

        <View style={styles.inputContainer}>
          <TextInput
            style={styles.input}
            onChange={(text) => this.setState({ title: text })}
            value={this.state.title}
            placeholder="Title"
            placeholderTextColor="white"
          />
        </View>
      </View>
    );
  }
}

// lanjutan kode ada di https://gist.github.com/aldipee/3ddded66ad5bd546e25b5a7bdfcb2950
```



Kemudian di dalam folder containers/todos/ buatlah sebuah file dengan nama styles.js. Isi file ini sebagai kode-kode untuk membuat styling.

Nah, full syntax kode styling dapat kamu lihat pada [link ini](#) ya!

```
import { StyleSheet } from 'react-native';

const styles = StyleSheet.create({
  root: {
    paddingHorizontal: 12
  },
  headingContainer: {
    marginTop: 12,
    marginBottom: 22
  },
  heading: {
    fontSize: 24,
    color: 'white'
  },
  content: {
    marginTop: 20
  }
})
// ... kode styling lanjut
```



Pada file **App.js** yang terletak di bagian root folder project kamu, ubahlah syntax kode seperti gambar di samping.

```
import React from 'react';
import {
  SafeAreaView,
  StyleSheet
} from 'react-native';
import { Provider } from 'react-redux';

import { Store } from './Store';

import Todos from './Containers/Todos';

const styles = StyleSheet.create({
  root: {
    backgroundColor: '#263238',
    flex: 1,
    flexDirection: 'column',
  }
});

const App = () => {

  const backgroundStyle = {
    backgroundColor: '#FFFFF',
  };

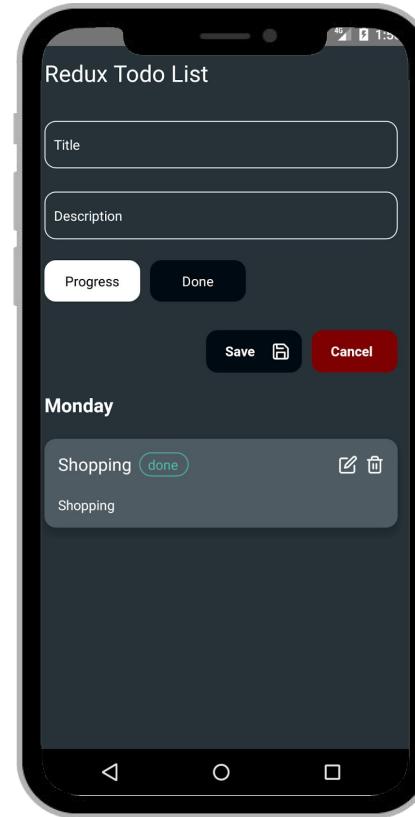
  return (
    <Provider store={Store}>
      <SafeAreaView style={styles.root}>
        <Todos />
      </SafeAreaView>
    </Provider>
  );
};

export default App;
```



Selanjutnya, jalankan ulang/reload project react native kamu, sehingga tampilan aplikasi kamu akan tampak seperti gambar di samping.

Nah sekarang kamu telah berhasil melakukan setup awal Redux dan elemen-elemennya (actions, reducers, types store).





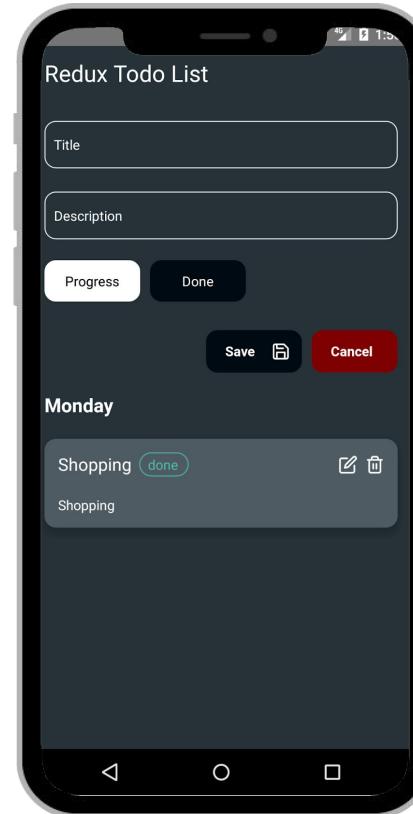
Nah, setelah berhasil setup awal dan bikin layout aplikasinya, kita lanjut lagi ya!  
Cekidot~





## HTTP Request menggunakan Redux-Thunk

Langkah selanjutnya, kamu akan belajar dan mencoba bagaimana proses HTTP Request menggunakan action Redux-Thunk, menyimpan data hasil dari HTTP Request tersebut ke dalam reducer, dan menampilkan data tersebut pada tampilan UI aplikasi react native kita.





## Install Axios dulu~

Nah, sekarang kita coba melakukan HTTP Request pada Redux-Thunk yuks.

Untuk bikin HTTP Request kamu akan memerlukan sebuah library **axios** sebagai HTTP Client yang bertugas membuat HTTP Request ke server pada aplikasi react native.

Untuk menginstall **axios** cukup ketik perintah `npm install axios` pada terminal.



```
npm install axios
```



## Next >> membuat function untuk mengambil data pada server

Pada file **index.js** yang ada di folder **actions**, buatlah sebuah function dengan nama **getTodos**.

Pada function `getTodos`, kamu akan memanggil API menggunakan `axios`. Ketika sudah mendapatkan datanya, kamu akan melakukan `dispatch()` action `saveTodos()` untuk mengupdate data pada Reducer dan menyimpannya pada Redux.

```
// file actions/index.js

import axios from 'axios';
import { FETCH_TODO_SUCCESS } from '../Types';

export const saveTodos = (data) => ({
  type: FETCH_TODO_SUCCESS,
  payload: data,
});

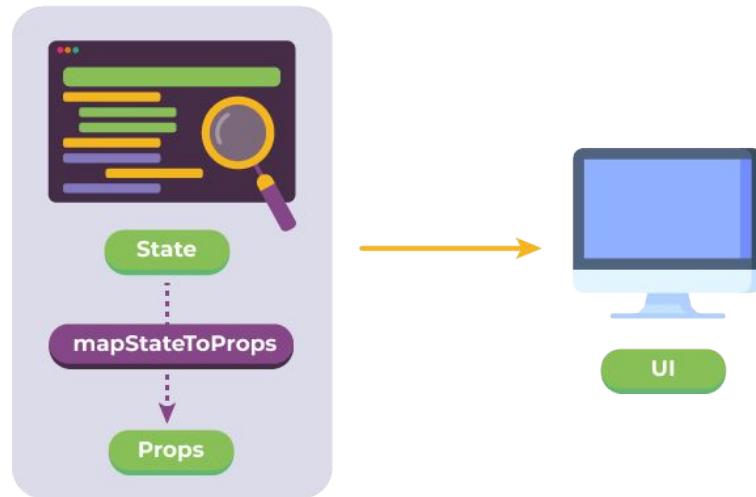
// function untuk mendapatkan data todos
export const getTodos = () => {
  return async (dispatch) => {
    const resTodos = await axios.get('http://code.aldipee.com/api/v1/todos');
    if (resTodos.data.results.length > 0) {
      dispatch(saveTodos(resTodos.data.results));
    }
  };
};
```



## Lalu menghubungkan component dengan Redux

Ketika kamu menggunakan Redux sebagai state management, kamu memerlukan cara khusus untuk dapat menggunakan State tersebut ke dalam UI yang kamu buat.

Bagaimana caranya? Simak di slide selanjutnya yuk~





1. Gunakan **mapStateToProps**. Fungsi ini berguna untuk mengambil State pada Redux, yang kemudian bisa kamu gunakan sebagai **props**.

```
● ● ●
import React from 'react';
import {
  View,
  TouchableOpacity,
  Text,
  TextInput
} from 'react-native';
import Icon from 'react-native-vector-icons/Feather';

import { connect } from 'react-redux';

import styles from './styles';

class Todos extends React.Component {

  constructor(props) {
    super(props);
    this.state = {
      title: '',
      description: '',
      selectedStatus: 'ON_PROGRESS',
    }
  }

  render() {
    return (
      <View style={styles.root}>
        // Kode Komponen Todos
      </View>
    )
  }

}

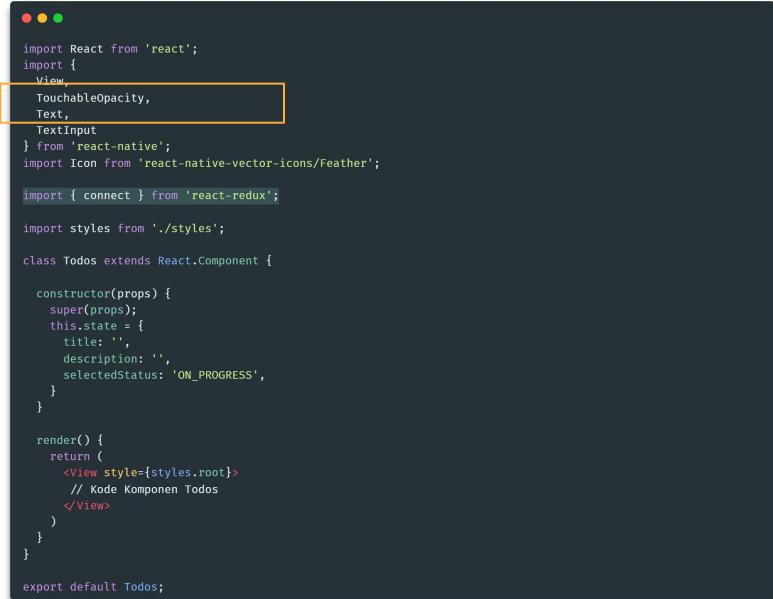
export default Todos;
```



2. Pada file komponen Todos yang terletak pada container/todos/index.js, import method connect dari package react-Redux dengan cara:

```
import { connect } from 'react-Redux';
```

Method connect inilah yang berfungsi untuk membuat koneksi antara komponen Todos dengan Redux Store.



```
import React from 'react';
import {
  View,
  TouchableOpacity,
  Text,
  TextInput
} from 'react-native';
import Icon from 'react-native-vector-icons/Feather';

import { connect } from 'react-redux';

import styles from './styles';

class Todos extends React.Component {

  constructor(props) {
    super(props);
    this.state = {
      title: '',
      description: '',
      selectedStatus: 'ON_PROGRESS',
    }
  }

  render() {
    return (
      <View style={styles.root}>
        // Kode Komponen Todos
      </View>
    )
  }
}

export default Todos;
```



3. Kemudian import action yang telah kamu bikin sebelumnya. Action getTodos pada file actions/index.js :

```
import { getTodos } from './actions';
```

```
● ● ●

import React from 'react';
import {
  View,
  TouchableOpacity,
  Text,
  TextInput
} from 'react-native';
import Icon from 'react-native-vector-icons/Feather';

import { connect } from 'react-redux';

import styles from './styles';
import { getTodos } from '../..//actions';

class Todos extends React.Component {

  constructor(props) {
    super(props);
    this.state = {
      title: '',
      description: '',
      selectedStatus: 'ON_PROGRESS',
    }
  }

  render() {
    return (
      <View style={styles.root}>
        // Kode Komponen Todos
      </View>
    )
  }

}

export default Todos;
```



4. Setelah itu, pada dua variable di bagian akhir component (sebelum syntax export default), buatlah dua variable berikut:

#### **mapStateToProps** dan **mapDispatchToProps**.

Kedua variable ini nantinya akan dimasukan sebagai parameter pada method connect() yang akan membuat koneksi antara komponen dengan Redux store.

```
import React from 'react';
import {
  View,
  TouchableOpacity,
  Text,
  TextInput
} from 'react-native';
import Icon from 'react-native-vector-icons/Feather';

import { connect } from 'react-redux';

import styles from './styles';
import { getTodos } from '../../actions'

class Todos extends React.Component {

  constructor(props) {
    super(props);
    this.state = {
      title: '',
      description: '',
      selectedStatus: 'ON_PROGRESS',
    }
  }

  render() {
    return (
      <View style={styles.root}>
        // Kode Komponen Todos
      </View>
    )
  }
}

const mapStateToProps = (state) => {
  return {
    data: state.appdata.todos
  };
};

const mapDispatchToProps = {
  getTodoData: getTodos
};

export default Todos;
```

Pada bagian  
**mapDispatchToProps**,  
masukan function  
**getTodos** yang telah  
kita import dari folder  
actions



5. Setelah itu panggil method connect pada bagian export default, dengan memasukan dua parameter yang berasal dari variable **mapStateToProps** dan **mapDispatchToProps**.



```
import React from 'react';
import {
  View,
  TouchableOpacity,
  Text,
  TextInput
} from 'react-native';
import Icon from 'react-native-vector-icons/Feather';

import { connect } from 'react-redux';

import styles from './styles';
import { getTodos } from '../../actions';

class Todos extends React.Component {

  constructor(props) {
    super(props);
    this.state = {
      title: '',
      description: '',
      selectedStatus: 'ON_PROGRESS',
    }
  }

  render() {
    return (
      <View style={styles.root}>
        // Kode Komponen Todos
      </View>
    )
  }
}

const mapStateToProps = (state) => {
  return {
    data: state.appData.todos
  };
};

const mapDispatchToProps = {
  getTodoData: getTodos
};

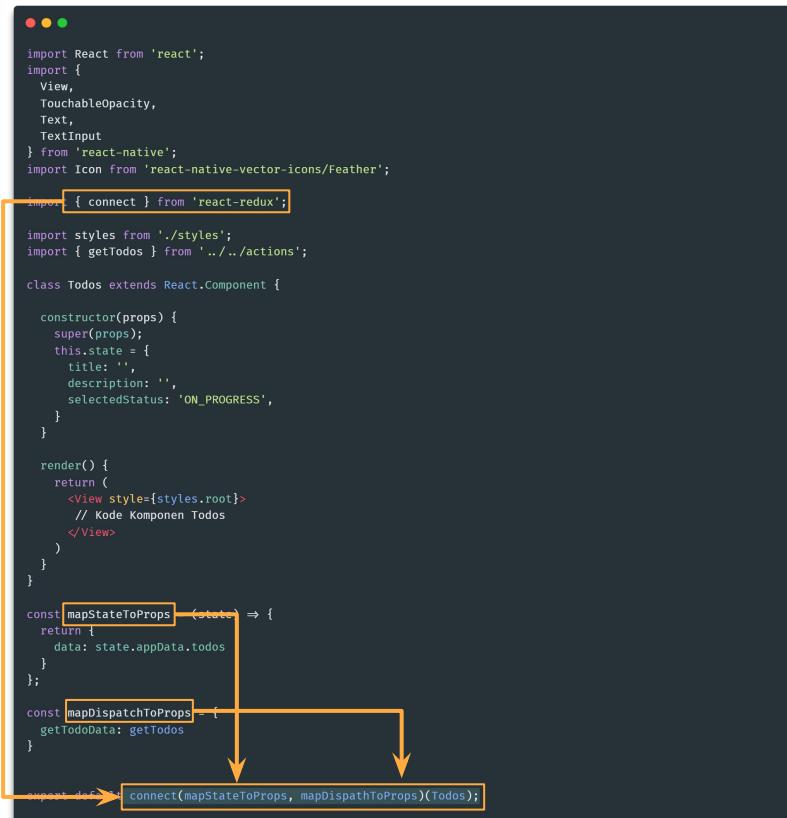
export default connect(mapStateToProps, mapDispatchToProps)(Todos);
```



## mapStateToProps dan mapDispatchToProps

**mapStateToProps** digunakan untuk mengakses State yang tersimpan pada Redux ke dalam component, sedangkan **mapDispatchToProps** digunakan untuk mengakses Redux Action dari component.

Dengan begitu, kamu bisa melakukan dispatch Action langsung dari component yang sudah kamu buat.



```
import React from 'react';
import {
  View,
  TouchableOpacity,
  Text,
  TextInput
} from 'react-native';
import Icon from 'react-native-vector-icons/Feather';

import { connect } from 'react-redux';

import styles from './styles';
import { getTodos } from '../../actions';

class Todos extends React.Component {

  constructor(props) {
    super(props);
    this.state = {
      title: '',
      description: '',
      selectedStatus: 'ON_PROGRESS',
    }
  }

  render() {
    return (
      <View style={styles.root}>
        // Kode Komponen Todos
      </View>
    )
  }
}

const mapStateToProps = (state) => {
  return {
    data: state.appData.todos
  };
};

const mapDispatchToProps = {
  getTodoData: getTodos
};

export default connect(mapStateToProps, mapDispatchToProps)(Todos);
```

The diagram illustrates the flow of the mapStateToProps and mapDispatchToProps functions. It starts with their definitions in the Todos component, which are then passed as arguments to the connect() function at the bottom. Arrows point from each function's definition to its corresponding argument in the connect() call.



## Ini bedanya kalau kita menggunakan Redux dan nggak menggunakan Redux

Dengan menggunakan Redux, fetching datanya pakai Action yang berada di dalam props.



```
componentDidMount(){
  this.props.getTodoData()
}
```

Tanpa menggunakan Redux, fetching data manual pake axios.



```
componentDidMount(){
  axios.get('http://code.aldipee/api/v1/todos').then(data =>{
    console.log(data)
  })
}
```



Eh tapi, `this.props.getTodoData()` dapet dari mana ya?

Kan saat kita panggil, component `<Todo />` nya nggak masukin props?



```
componentDidMount(){
    this.props.getTodoData()
}
```



Kamu ingat nggak? Sebelumnya, pada kode komponen Todos kita mendefinisikan sebuah object.

Nah, setiap object yang kita definisikan di variabel **mapStateToProps** dan **mapDispatchToProps** nantinya bakal bisa langsung di akses di dalam this.props, guys.

```
const mapDispatchToProps = {  
  getTodoData: getTodos  
}
```

```
componentDidMount(){  
  this.props.getTodoData()  
}
```



Karena kamu sudah menulis `getTodoData` di dalam `mapDispatchToProps`, maka `getTodoData` ini bisa kamu akses dengan `this.props.getTodoData`.

Begini pula kalau kamu menambahkan Action lain di dalam kedua variabel itu, semuanya akan dapat diakses melalui props pada component tersebut.



```
const mapDispatchToProps = {
  getTodoData: getTodos
}
```



```
componentDidMount(){
  this.props.getTodoData()
}
```



Sama halnya pada variable **mapStateToProps**, semua property yang didefinisikan pada bagian return di variable **mapStateToProps**, nantinya bakal dapat kamu akses di dalam props.

Pada kasus di component Todos, kamu memberi property data di dalam **mapStateToProps**, maka nantinya saat kamu mau mengakses data item todo list, kamu bisa pakai props dengan cara **this.props.data**

```
const mapStateToProps = (state) => {
  return {
    data: state.appData.todos
  }
};

const mapDispatchToProps = {
  getTodoData: getTodos
}
```

```
console.log(this.props.data)
```



## Perbedaan `mapStateToProps` dan `mapDispatchToProps`, apa hayoo?

Perbedaan `mapStateToProps` dan `mapDispatchToProps` ada pada isinya.

`MapDispatchToProps` berisi **function action**, sedangkan `mapStateToProps` berisi **cara mengakses data pada Redux Store** agar bisa di akses pada variabel `this.props`

```
const mapStateToProps = (state) => {
  return {
    data: state.appData.todos
  }
};

const mapDispatchToProps = {
  getTodoData: getTodos
}
```

```
this.props.getTodoData()
console.log(this.props.data)
```



## Saatnya belajar cara render data ke UI~

To the point aja. Gimana sih caranya?

1. Begini guys, component Todos yang menampilkan tampilan list kan saat ini masih static.

Karena component Todos kita telah terkoneksi ke Redux, maka kamu bisa langsung melakukan perulangan data tersebut dari variabel this.props.data.

```
render() {
  return (
    // ...kode jsx lainnya
    <View style={styles.content}>
      <Text style={styles.todoDateText}>Monday</Text>
      <View style={styles.cardListContainer}>
        <View style={styles.todoCard}>
          <View style={styles.todoTitleContainer}>
            <View style={styles.todoActionContainer}>
              <Text style={styles.todoTitle}>Shopping</Text>
              <TouchableOpacity style={styles.doneBadge}>
                <Text style={styles.doneBadgeText}>done</Text>
              </TouchableOpacity>
            </View>
            <View style={styles.todoActionContainer}>
              <TouchableOpacity style={styles.editButton}>
                <Icon name="edit" size={20} color="white" />
              </TouchableOpacity>
              <TouchableOpacity>
                <Icon name="trash-2" size={20} color="white" />
              </TouchableOpacity>
            </View>
          </View>
          <View>
            <Text style={styles.todoDescription}>Shopping</Text>
          </View>
        </View>
      </View>
    );
}
```



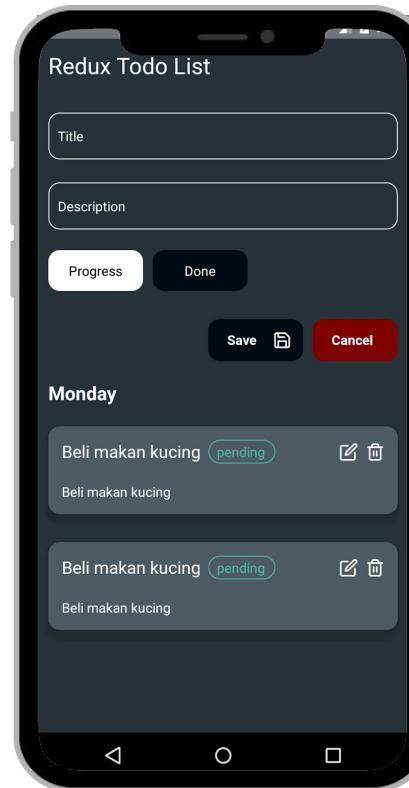
1. Pertama, kita membuat komponen yang akan menampilkan data. Komponen ini akan menerima properti `this.props.data` dan akan mengembalikan JSX yang menampilkan data.
2. Kemudian, maping variabel `this.props.data` menggunakan method `.map()` untuk menampilkan semua data object yang terdapat di dalam array, yang tersimpan di variable `this.props.data`.

```
render() {
  return (
    // ... kode jsx lainnya
    <View style={styles.content}>
      <Text style={styles.todoDateText}>Monday</Text>
      {this.props.data.map(itemTodo => (
        <View style={styles.cardListContainer}>
          <View style={styles.todoCard}>
            <View style={styles.todoTitleContainer}>
              <View style={styles.todoActionContainer}>
                <Text style={styles.todoTitle}>{itemTodo.title}</Text>
                <TouchableOpacity style={styles.doneBadge}>
                  <Text style={styles.doneBadgeText}>{itemTodo.status}</Text>
                </TouchableOpacity>
              </View>
              <View style={styles.todoActionContainer}>
                <TouchableOpacity style={styles.editButton}>
                  <Icon name="edit" size={20} color="white" />
                </TouchableOpacity>
                <TouchableOpacity style={styles.deleteButton}>
                  <Icon name="trash-2" size={20} color="white" />
                </TouchableOpacity>
              </View>
            </View>
            <Text style={styles.todoDescription}>{itemTodo.title}</Text>
          </View>
        </View>
      )));
    </View>
  );
}
```



3. Sekarang coba reload atau running ulang aplikasi react native-mu. Tampilan list-nya pasti akan berubah jadi lebih dinamis dan berhasil mengambil data dari server.

Selain itu, kamu juga berhasil mengintegrasikan API dengan menggunakan cara Redux pada aplikasi react native.





Di Redux juga ada package bernama **Redux Persist** yang bakal kamu butuhkan banget ketika membuat aplikasi

Buat apa sih Redux persist ini? Cuss kita kepoin~





## Kenalan dulu deh sama Redux Persist

Seperti yang udah dijelaskan sebelumnya, Redux merupakan sebuah state yang bersifat global.

Sebagaimana State pada umumnya, **data pada Redux nggak bersifat permanen**, artinya ketika aplikasi kamu close lalu kamu buka lagi atau direload, maka data yang tersimpan pada Redux juga akan hilang.

Nah tapiii...





Ada beberapa kasus pada aplikasi di mana kamu akan membutuhkan State pada Redux untuk tetap ada walaupun aplikasi sudah kita tutup/reload.

Contohnya penyimpanan token autentikasi, menambahkan fungsionalitas ketika aplikasi sedang offline, dan lain-lain.





## Kenalin nih, Redux-persist

Untuk itulah, kamu butuh package bernama Redux-persist. Dengan package ini, kamu bisa **menentukan State mana yang akan disimpan pada storage aplikasi.**





### Eits, tunggu dulu ,Sob..

Tetapi kamu juga perlu tahu nih, bahwa **nggak semua State perlu dimasukkan ke dalam Redux-persist.**

Kenapa?

karena Redux-persist akan menyimpan state ke dalam Storage Phone Aplikasi dan kalau kebanyakan tentu akan mempengaruhi performa aplikasi yang jadi lebih lambat.





### Redux persist perlu diintegrasikan dengan aplikasi ya, guys!

Untuk mengintegrasikan aplikasi dengan Redux-persist, ada dua library yang harus kamu install pada project react native yaitu:

- [\*\*@react-native-async-storage/async-storage\*\*](#)
- **Redux-persist.**

redux-persist



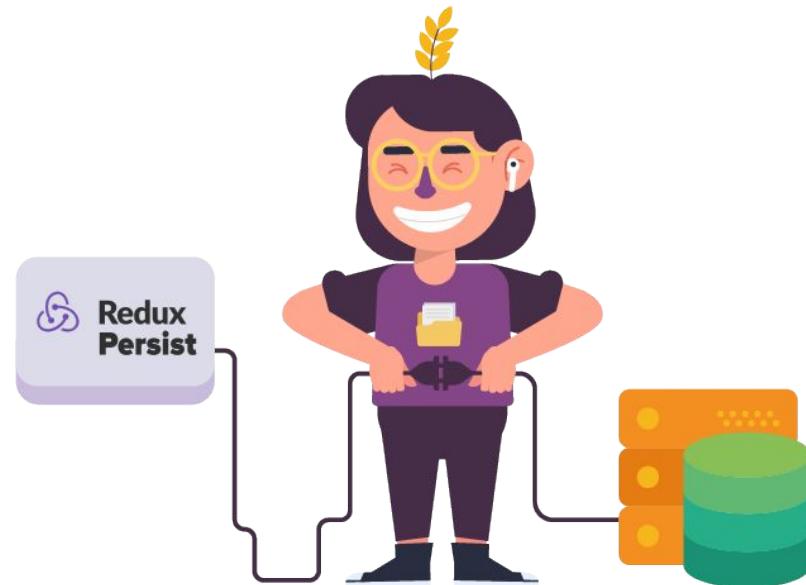
async-storage





### FYI, ini dia fungsi kedua library~

`@react-native-async-storage/async-storage` berfungsi sebagai **jembatan yang akan menghubungkan** Redux-persist dengan memori penyimpanan yang terdapat pada Android/iOS kamu.





Langsung aja kita mulai langkah-langkah  
untuk menggunakan Redux persist~





## Pertama, Install React Native Async Storage dulu ya~

Untuk mulai menggunakan @react-native-async-storage/async-storage kita cukup menginstall package nya melalui npm dengan cara mengetik command:

**npm install @react-native-async-storage/async-storage**

Kemudian tunggu hingga proses instalasi selesai.

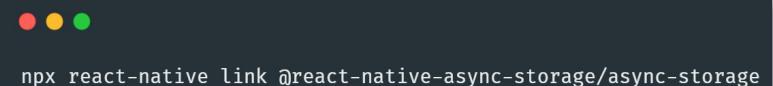
```
npm install @react-native-async-storage/async-storage
```



Kemudian lakukan link agar native dependencies yang terdapat pada **@react-native-async-storage/async-storage** terinstall di project kamu.

Ketik command :

```
npx react-native link  
@react-native-async-storage/async-storage
```



```
npx react-native link @react-native-async-storage/async-storage
```



### Untuk iOS perlu install ini ya

Kemudian **install juga dependency pods** yang dibutuhkan oleh platform iOS dengan cara mengetikan command :

**npx pod-install**



```
npx pod-install
```



### Lanjut dengan menginstal Redux Persist

Untuk mulai menggunakan **Redux-persist** kita cukup menginstal package nya melalui npm dengan cara mengetik command:

**npm install Redux-persist**

Kemudian tunggu hingga proses instalasi selesai.



**npm install redux-persist**



## Kemudian edit file store/index.js

Pada file konfigurasi Redux yang terdapat pada folder **store** dan file dengan nama **index.js**, tambahkan beberapa konfigurasi tambahan buat Redux persist. Caranya di bawah ini ya!

**1. Import Redux-persist dan Async Storage**

**2. Definisikan konfigurasi Redux persist**

**3. Kemudian buat sebuah variable dengan nama Persistor, dan export variable tersebut**

```
import {applyMiddleware, combineReducers, createStore } from 'redux';
import ReduxThunk from 'redux-thunk';

import TodosReducer from '../Reducers';

const Reducers = {
  appData: TodosReducer
}

export const Store = createStore(combineReducers(TodosReducers), applyMiddleware(ReduxThunk))
```

Kondisi file store/index.js sekarang

```
import {applyMiddleware, combineReducers, createStore } from 'redux';
import ReduxThunk from 'redux-thunk';

import AsyncStorage from '@react-native-community/async-storage';
import { persistStore, persistReducer } from 'redux-persist';

import TodosReducer from '../Reducers';

const Reducers = {
  appData: TodosReducer
}

const persistConfig = {
  key: 'root',
  storage: AsyncStorage
}
const configPersist = persistReducer(persistConfig, combineReducers(TodosReducers))

export const Store = createStore(configPersist, applyMiddleware(ReduxThunk))
export const Persistor = persistStore(Store)
```



## Lalu Edit file App.js yang terdapat pada root folder project.

Caranya di bawah ini ya!

1. Import component **PersistGate** dari package '**Redux-persist/integration/react**', dan variable **Persistor** yang kita telah buat sebelumnya dari file Store/index.js

2. Pada bagian ini kita menambahkan component **<PersistGate>** yang akan menjadi jalan masuk utama bagi Redux-persist untuk mendapatkan akses ke State aplikasi kita.

```
import React from 'react';
import {
  SafeAreaView,
  StyleSheet
} from 'react-native';
import { Provider } from 'react-redux';
import { PersistGate } from 'redux-persist/integration/react';
import { Store, Persistor } from './Store';

import Todos from './Containers/Todos';

const styles = StyleSheet.create({
  root: {
    backgroundColor: '#263238',
    flex: 1,
    flexDirection: 'column',
  }
});

const App = () => {

  const backgroundStyle = {
    backgroundColor: '#FFFFFF',
  };

  return (
    <Provider store={Store}>
      <PersistGate loading={null} persistor={Persistor}>
        <SafeAreaView style={styles.root}>
          <Todos />
        </SafeAreaView>
      </PersistGate>
    </Provider>
  );
};

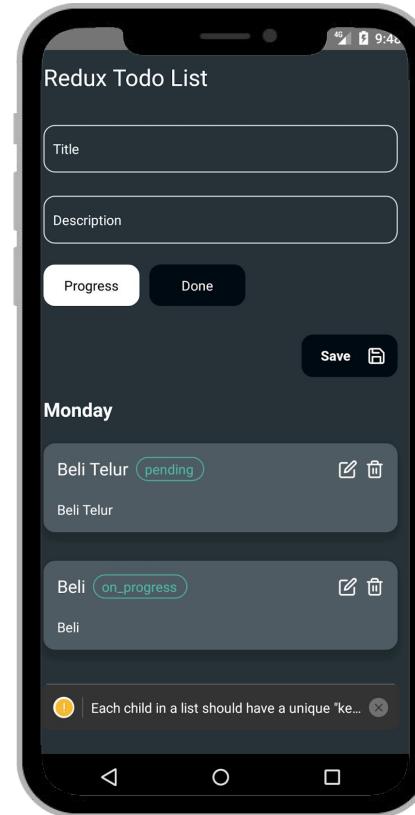
export default App;
```



## Terakhir, running ulang project react native

Nah, reload/running ulang project aplikasi react native kamu.

Sekarang Redux-persist telah berhasil terintegrasi pada aplikasi react native kamu.



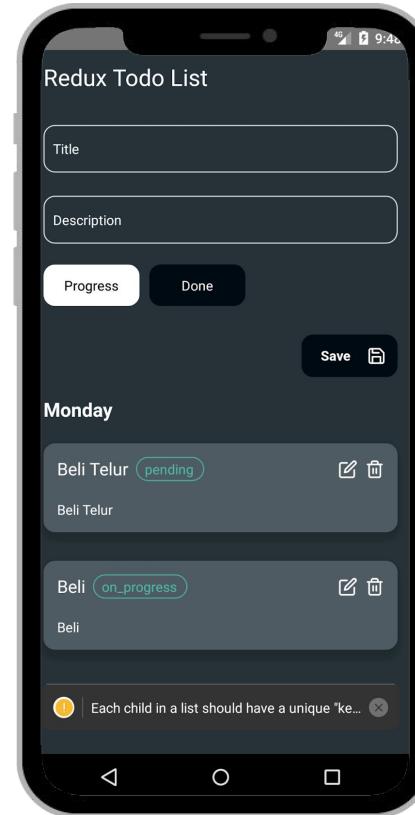


## Mari kita tes keberhasilan Redux persist

Ikuti langkah berikut ya!

1. Buka aplikasi
2. Tutup lagi aplikasimu
3. Putuskan koneksi internet
4. Buka lagi aplikasinya

Kesimpulannya adalah... >>next slide yah

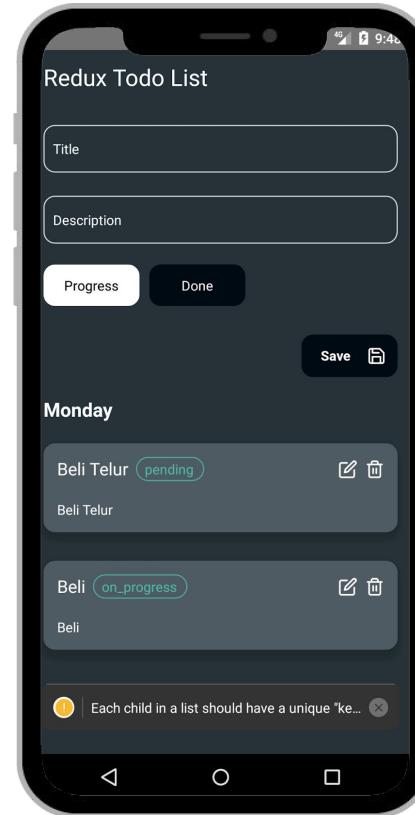




### Begini tandanya kalau Redux persist berhasil

Jika Redux persist berhasil, maka secara otomatis aplikasi react native kamu akan tetap menampilkan data berdasarkan hasil HTTP Request yang berhasil sebelumnya.

Dengan ini secara otomatis aplikasi kamu tetap akan menampilkan data walaupun user tidak terkoneksi dengan internet.



# Saatnya kita Quiz!





## 1. Apa fungsi dari middleware pada Redux?

- A. Penambahan sebuah proses setelah action ke trigger dan sebelum reducer ke update
- B. Proses duplikasi data pada reducer sebelum disimpan pada store
- C. Untuk menyimpan data pada memori aplikasi



## 1. Apa fungsi dari middleware pada Redux?

- A. Untuk menambahkan sebuah proses setelah action ke trigger dan sebelum reducer ke update
- B. Proses duplikasi data pada reducer sebelum disimpan pada store
- C. Untuk menyimpan data pada memori aplikasi

Redux Middleware berfungsi untuk sebuah proses tambahan yang dilakukan sebelum proses reducer dan setelah proses trigger sebuah action



## 2. Bagaimana cara menambahkan lebih dari satu middleware pada method applyMiddleware?

- A. applyMiddleware(middleware1, middleware2, middleware3)
- B. applyMiddleware([middleware1, middleware2, middleware3])
- C. applyMiddleware({middleware1, middleware2, middleware3})



## 2. Bagaimana cara menambahkan lebih dari satu middleware pada method applyMiddleware?

- A. applyMiddleware(middleware1, middleware2, middleware3)
- B. applyMiddleware([middleware1, middleware2, middleware3])
- C. applyMiddleware({middleware1, middleware2, middleware3})

Method applyMiddleware dapat menerima banyak parameter.  
Tapi parameternya bukan berbentuk array ataupun object  
Lengkapnya cek <https://Redux.js.org/api/applyMiddleware>



### 3. Data yang disimpan pada Redux store bersifat...

- A. Immutable
- B. Mutable
- C. Keduanya benar



### 3. Data yang disimpan pada Redux store bersifat...

- A. **Immutable**
- B. Mutable
- C. Keduanya benar

**Data pada Redux store bersifat immutable atau tidak dapat diubah secara langsung**



**4. Daniela ingin membuat sebuah aplikasi yang mana ketika user telah login sekali, maka ketika selanjutnya aplikasi dibuka si user tidak perlu login lagi, untuk membuat fitur ini daniela harus menggunakan Redux-persist, data apa saja yang harus disimpan oleh daniela pada Redux-persist?**

- A. Data informasi login, token, profile dan data yang bersifat nya static (tidak berubah-ubah).
- B. Data konten dari aplikasi tersebut, sehingga apabila ketika internet offline konten dapat di akses.
- C. Data username dan password, sehingga ketika user login kedua kalinya akan ada pemanggilan otomatis api untuk login dengan informasi username dan password yang disimpan pada Redux-persist



**4. Daniela ingin membuat sebuah aplikasi yang mana ketika user telah login sekali, maka ketika selanjutnya aplikasi dibuka si user tidak perlu login lagi, untuk membuat fitur ini daniela harus menggunakan Redux-persist, data apa saja yang harus disimpan oleh daniela pada Redux-persist?**

- A. **Data informasi login, token, profile dan data yang bersifat nya static (tidak berubah-ubah).**
- B. Data konten dari aplikasi tersebut, sehingga apabila ketika internet offline konten dapat diakses.
- C. Data username dan password, sehingga ketika user login kedua kalinya akan ada pemanggilan otomatis api untuk login dengan informasi username dan password yang disimpan pada Redux-persist

**Data yang harus disimpan adalah berubah informasi login, token access, dan profile.**

# Terima Kasih!



Chapter ✓

completed