



Redux State Management 2

Gold - Chapter 4 - Topic 2

**Selamat datang di Chapter 4 Topic 2 online
course React Native dari Binar Academy!**

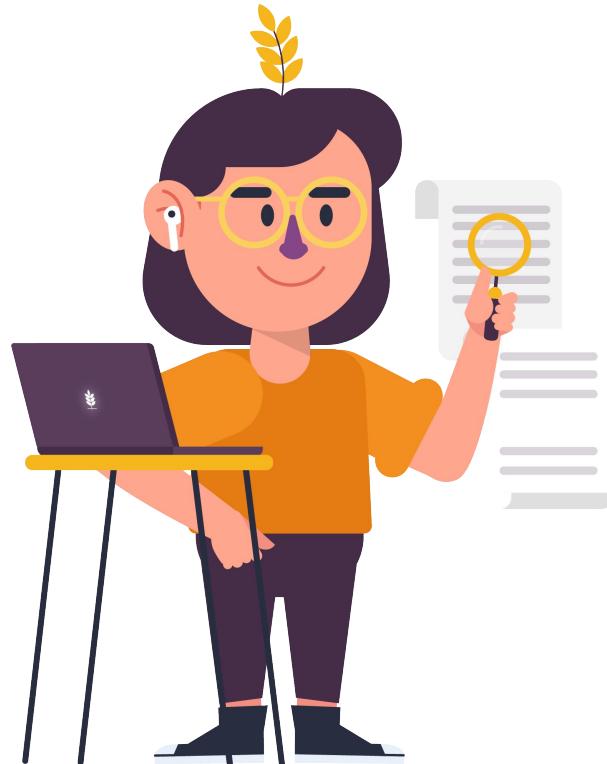




Pergi ke pesta pagi-pagi Haiii, kita ketemu lagi! 🙌

Di topic 1 kita sudah bahas bagian permukaan dari Redux. Jadi di topic 2 ini kita bakal bahas lebih jauh lagi soal Redux State Management ya, guys!

Ayo kita mulai!



Detailnya, kita bakal bahas hal-hal berikut ini:

- Integrasi UI dan Redux
- Debugging with Redux Logger





Guys, masih ingat kan?

Data-data yang ada di sebuah state harus dibuat tampilan UI-nya biar tetep kece kalau dilihat pengguna.





Nah, karena di topic sebelumnya kita sudah bahas Redux Store, maka kita juga harus tahu cara bikin tampilan UI yang terkoneksi dengan redux store

Mari kita lanjut~





Sebelum mulai, kita balik dikit ke asal usul Redux

Bayangin kamu bikin project besar dengan menggunakan React atau React Native. Kamu punya ratusan komponen aplikasi dan di setiap komponen tersebut punya data (state) yang harus digunakan di banyak komponen.





Lanjutannya~

Kalau dijalankan sesuai prinsip state —yang cuma bisa dijalankan di satu komponen, maka akan ribet banget kalau kamu mengirim state langsung dari komponen dan project yang kamu buat.

Bukan cuma ribet, kamu juga bakal kesulitan dalam membaca dan melakukan testing projectmu.

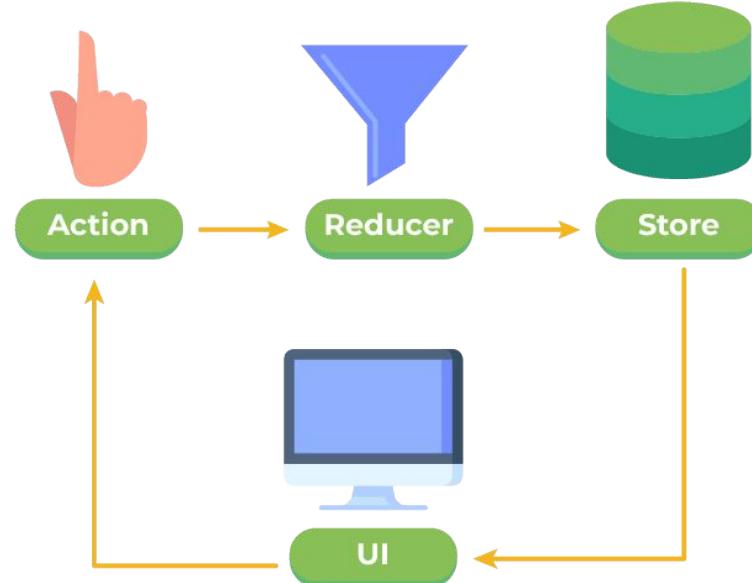
Karena itu muncullah Redux. Dengan Redux semua State/Data/Props/Function **dapat digunakan di semua Class/Component/Screen yang kamu buat.**



Terus bagaimana cara kerja Redux?

Ketika User melakukan Action -> maka akan mentrigger State pada Reducer -> setelah itu Data di Reducer disimpan ke Root State Global (Store) -> Dan terakhir akan ditampilkan kepada user.

Nah **apabila UI aplikasimu telah terkoneksi dengan store**, maka setiap ada perubahan data di Store, data yang ditampilkan pada UI akan berubah juga.



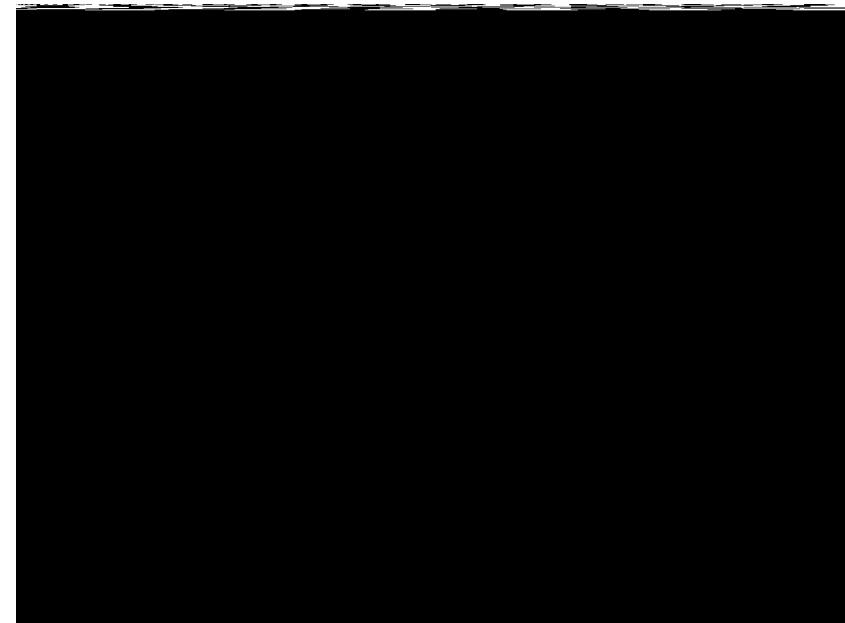


Biar lebih kebayang, coba kita pakai contoh ilustrasi deh..

Ketika **button Deposit \$10** diklik, sebuah function akan Dispatch (aktif) dan meneruskan sebuah action dengan **type: deposit** dan **payload: 10**. Payload adalah value yang ada di dalam action.

Kemudian Redux akan mengeksekusi function reducer yang sesuai dengan action yang diteruskan.

Reducer akan mengubah state dari \$0 ke \$10 untuk kemudian ditampilkan di sisi UI.



[KLIK DI SINI](#)



Setelah paham tentang cara kerja Redux, pertanyaan selanjutnya adalah bagaimana ya cara bikin tampilan UI yang terkoneksi dengan Redux Store?



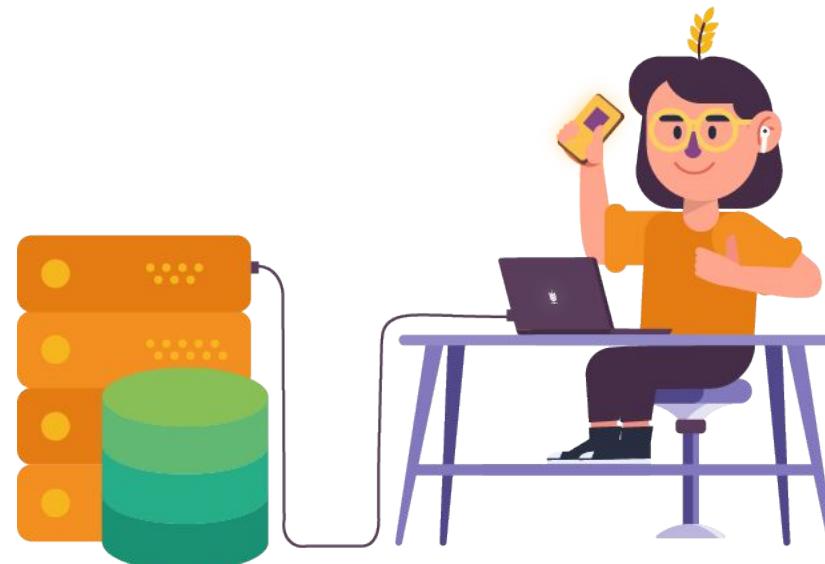


Oke, mari kita pelajari!

Bagaimana membuat tampilan UI yang terkoneksi dengan redux store?

Caranya adalah kamu akan **membuat sebuah aplikasi simple counter** yang akan menampilkan data dari redux store. Selain itu, kamu juga akan **membuat sebuah trigger action** yang akan mengupdate data counter setiap kali button plus atau minus diklik.

Cuss, mulai!



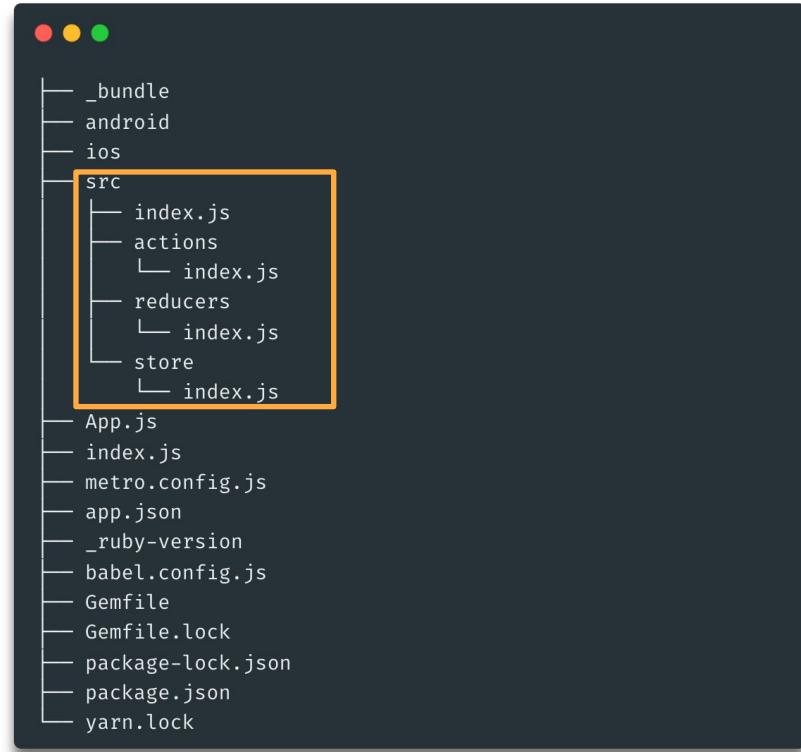


1. Buat struktur folder

Pada project kali ini kamu akan bikin beberapa folder untuk menyimpan element dari Redux.

Buatlah sebuah folder dengan nama **src** pada root project, kemudian di dalam folder src tersebut, buatlah folder **actions**, **reducers** dan **store**.

Terakhir, di dalam setiap folder tersebut buatlah file index.js. Kemudian tambahkan file index.js di dalam folder src.



```
_bundle
android
ios
src
├── index.js
├── actions
│   └── index.js
├── reducers
│   └── index.js
└── store
    └── index.js
App.js
index.js
metro.config.js
app.json
_ruby-version
babel.config.js
Gemfile
Gemfile.lock
package-lock.json
package.json
yarn.lock
```



2. Buat tampilan UI

Pada file index.js yang terdapat di dalam folder src, buatlah sebuah component Counter.

Komponen ini nantinya yang akan menampilkan tampilan counter pada aplikasi kamu.

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

class Counter extends React.Component {
  render() {
    return (
      <View style={styles.counterContainer}>
        <Text style={styles.counterInfo}>HITUNG: 0</Text>
        <View style={styles.counterBtnContainer}>
          <TouchableOpacity style={styles.counterButtonMinus}>
            <Text style={styles.counterText}>KURANG (-)</Text>
          </TouchableOpacity>
          <TouchableOpacity style={styles.counterButtonPlus}>
            <Text style={styles.counterText}>TAMBAH (+)</Text>
          </TouchableOpacity>
        </View>
      </View>
    );
  }
}

export default Counter;
```



Untuk kode-kode styling komponen Counter, bisa kamu tulis berdasarkan syntax di samping.

Kamu boleh menulis styling ini pada file yang sama dengan komponen atau bikin file style terpisah dengan komponen.

```
const styles = StyleSheet.create({
  counterContainer: {
    width: '100%',
    padding: 20,
    backgroundColor: 'white',
    shadowColor: '#000',
    shadowOffset: {
      width: 4,
      height: 3,
    },
    shadowOpacity: 0.32,
    shadowRadius: 5.46,
    elevation: 9,
    marginBottom: 10,
  },
  counterInfo: {
    justifyContent: 'center',
    alignItems: 'center',
    textAlign: 'center',
    fontSize: 18,
  },
  counterBtnContainer: {
    flexDirection: 'row',
    marginTop: 20,
    width: '100%',
  },
  counterButtonPlus: {
    backgroundColor: '#90db7d',
    marginLeft: 5,
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  counterButtonMinus: {
    backgroundColor: '#db6767',
    marginLeft: 5,
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  counterText: {
    color: '#0e0e0e',
    padding: 10,
  },
});
```



Nah, tampilan yang akan muncul pada aplikasi kamu akan kayak gambar di samping nih.



3. Instal Redux

Langkah selanjutnya adalah menginstal redux pada project react native. Untuk menginstall redux prosesnya cukup sederhana kok. Ketik perintah berikut pada terminal/cmd :

npm install redux react-redux

Setelah itu reload/running ulang project react native kamu.

>> Setelah menginstal Redux, langkah selanjutnya adalah membuat file elemen-elemen untuk Redux yaitu actions, reducers, dan store.



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



4. Buat Actions pada Redux

Pada file index.js yang terdapat di dalam folder /src/actions/, tambahkan dua actions yaitu '**tambahinAngka**' dan '**kuranginAngka**'.

Kedua function ini nantinya akan dipanggil ketika user mengklik tombol "**Tambah**" atau "**Kurang**"

```
// file src/actions/index.js

export const tambahinAngka = () => ({
  type: 'TAMBAH_ANGKA',
});

export const kuranginAngka = () => ({
  type: 'KURANG_ANGKA',
});
```



4. Buat Reducers

Pada file index.js yang terdapat di dalam folder /src/reducers/, buatlah sebuah reducer. Reducer ini nantinya akan menangkap type (ketikan) yang dikirim ketika action ke trigger.

```
// file /src/reducers/index.js

const initialState = {
  angkaSekarang: 0,
};

const angkaReducer = (state = initialState, action) => {
  switch (action.type) {
    case 'TAMBAH_ANGKA':
      return { ...state, angkaSekarang: state.angkaSekarang + 1 };
    case 'KURANG_ANGKA':
      return { ...state, angkaSekarang: state.angkaSekarang - 1 };
    default:
      return state;
  }
};

export default angkaReducer;
```



Apabila type yang dibawa oleh si action adalah 'TAMBAH_ANGKA', maka reducer akan menambah value state angkaSekarang dengan angka 1.

Sebaliknya, apabila type yang dibawa oleh si action adalah 'KURANG_ANGKA', maka reducer akan mengurangi nilai angkaSekarang dengan angka 1.

```
// file /src/reducers/index.js

const initialState = {
    angkaSekarang: 0,
};

const angkaReducer = (state = initialState, action) => {
    switch (action.type) {
        case 'TAMBAH_ANGKA':
            return { ...state, angkaSekarang: state.angkaSekarang + 1 };
        case 'KURANG_ANGKA':
            return { ...state, angkaSekarang: state.angkaSekarang - 1 };
        default:
            return state;
    }
};

export default angkaReducer;
```



5. Bungkus <Provider /> komponen

Import variabel store yang telah kamu export pada file `/src/store/index.js`, kemudian **masukan variabel store tersebut ke dalam props store yang terdapat pada component <Provider />**

Setelah itu reload/running ulang project react native kamu.

```
import React from 'react';
import {Provider} from 'react-redux';

import CounterComponent from './src/index';

import {store} from './src/store/index';

class App extends React.Component {
  render() {
    return (
      <Provider store={store}>
        <CounterComponent />
      </Provider>
    );
  }
}

export default App;
```



6. Menghubungkan komponen Dengan Redux

Kenapa sih harus dihubungin? Ya supaya data yang disimpan pada Redux Store nantinya bisa diakses oleh komponen Counter.

Langkah pertama yang harus kamu lakukan adalah **Import method connect** yang berasal dari library react-redux.

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

import {connect} from 'react-redux';

class Counter extends React.Component {

  render() {
    return (
      <View style={styles.counterContainer}>
        <Text style={styles.counterInfo}>
          HITUNG: 0
        </Text>
        <View style={styles.counterBtnContainer}>
          <TouchableOpacity
            style={styles.counterButtonMinus}>
            <Text style={styles.counterText}>KURANG (-)</Text>
          </TouchableOpacity>
          <TouchableOpacity
            style={styles.counterButtonPlus}>
            <Text style={styles.counterText}>TAMBAH (+)</Text>
          </TouchableOpacity>
        </View>
      </View>
    );
  }

  const mapStateToProps = state => {
    return {
      reducerAngka: state.angkaReducer,
    };
  };

  export default Counter;
```



Kemudian pada bagian bawah sebelum component tersebut di-export, **tambahkan sebuah variabel** yang berisikan sebuah arrow function. Lalu function ini akan menghasilkan sebuah object dengan properti `reducerAngka`.

Nah, value data yang terdapat pada store, nantinya akan bisa diakses melalui props dengan cara **this.props.reducerAngka**.

```
● ● ●

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

import {connect} from 'react-redux';

class Counter extends React.Component {

  render() {
    return (
      <View style={styles.counterContainer}>
        <Text style={styles.counterInfo}>
          HITUNG: 0
        </Text>
        <View style={styles.counterBtnContainer}>
          <TouchableOpacity
            style={styles.counterButtonMinus}>
            <Text style={styles.counterText}>KURANG (-)</Text>
          </TouchableOpacity>
          <TouchableOpacity
            style={styles.counterButtonPlus}>
            <Text style={styles.counterText}>TAMBAH (+)</Text>
          </TouchableOpacity>
        </View>
      </View>
    );
  }
}

const mapStateToProps = state => {
  return {
    reducerAngka: state.angkaReducer,
  };
};

export default Counter;
```



Selanjutnya, **koneksikan action yang telah kamu buat dengan komponen Counter.**

Action inilah yang nantinya akan mengupdate data pada reducer apabila kamu trigger.

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

import {connect} from 'react-redux';

import {tambahinAngka, kuranginAngka} from './actions/index';

class Counter extends React.Component {

  render() {
    return (
      <View style={styles.counterContainer}>
        <Text style={styles.counterInfo}>
          HITUNG: 0
        </Text>
        <View style={styles.counterBtnContainer}>
          <TouchableOpacity style={styles.counterButtonMinus}>
            <Text style={styles.counterText}>KURANG (-)</Text>
          </TouchableOpacity>
          <TouchableOpacity style={styles.counterButtonPlus}>
            <Text style={styles.counterText}>TAMBAH (+)</Text>
          </TouchableOpacity>
        </View>
      </View>
    );
  }
}

const mapStateToProps = state => {
  return {
    reducerAngka: state.angkaReducer,
  };
};

const mapDispatchToProps = {
  tambahinAngka,
  kuranginAngka,
};

export default Counter;
```



Selanjutnya, **panggil method connect yang telah kamu import** dengan memasukan dua parameter yang berupa variabel **mapStateToProps** dan **mapDispatchToProps**.

Hal yang perlu diperhatikan adalah mapStateToProps harus dimasukan pada parameter pertama pada method connect. Kurang lebih seperti ini syntaxnya :

**connect(mapStateToProps,
mapDispatchToProps)(NAMACOMPONENT)**

```
● ● ●

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

import {connect} from 'react-redux';
import {tambahinAngka, kuranginAngka} from './actions/index';

class Counter extends React.Component {
  render() {
    return (
      <View style={styles.counterContainer}>
        <Text style={styles.counterInfo}>
          HITUNG: 0
        </Text>
        <View style={styles.counterBtnContainer}>
          <TouchableOpacity
            style={styles.counterButtonMinus}>
            <Text style={styles.counterText}>KURANG (-)</Text>
          </TouchableOpacity>
          <TouchableOpacity
            style={styles.counterButtonPlus}>
            <Text style={styles.counterText}>TAMBAH (+)</Text>
          </TouchableOpacity>
        </View>
      </View>
    );
  }
}

const mapStateToProps = state => {
  return {
    reducerAngka: state.angkaReducer,
  };
};

const mapDispatchToProps = {
  tambahinAngka,
  kuranginAngka,
};

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



Nah, kalau sudah, coba **render text "Hitung:"** dengan variabel `this.props.reducerAngka.angkaSekarang`.

Variabel tersebut akan mengambil value yang terdapat pada Reducer.

```
● ● ●

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

import {connect} from 'react-redux';
import {tambahinAngka, kuranginAngka} from './actions/index';

class Counter extends React.Component {
  render() {
    return (
      <View style={styles.counterContainer}>
        <Text style={styles.counterText}>{this.props.reducerAngka.angkaSekarang}</Text>
        <Text style={styles.counterText}>HITUNG: {this.props.reducerAngka.angkaSekarang}</Text>
      </View>
    );
  }
}

const mapStateToProps = state => {
  return {
    reducerAngka: state.angkaReducer,
  };
};

const mapDispatchToProps = {
  tambahinAngka,
  kuranginAngka,
};

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



Langkah selanjutnya adalah coba ganti pada file index yang terdapat pada reducers.

Ganti properti angkaSekarang yang terdapat pada object initialState dengan angka 472.

Setelah itu save file tersebut.

```
const initialState = {
    angkaSekarang: 472,
};

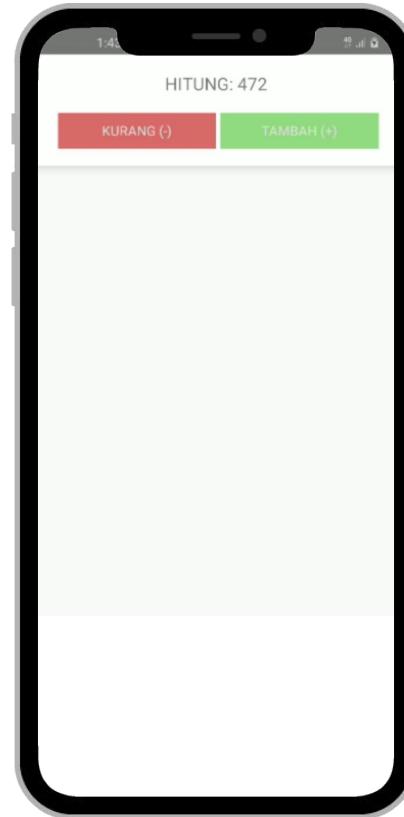
const angkaReducer = (state = initialState, action) => {
    switch (action.type) {
        case 'TAMBAH_ANGKA':
            return { ...state, angkaSekarang: state.angkaSekarang + 1};
        case 'KURANG_ANGKA':
            return { ...state, angkaSekarang: state.angkaSekarang - 1};
        default:
            return state;
    }
};

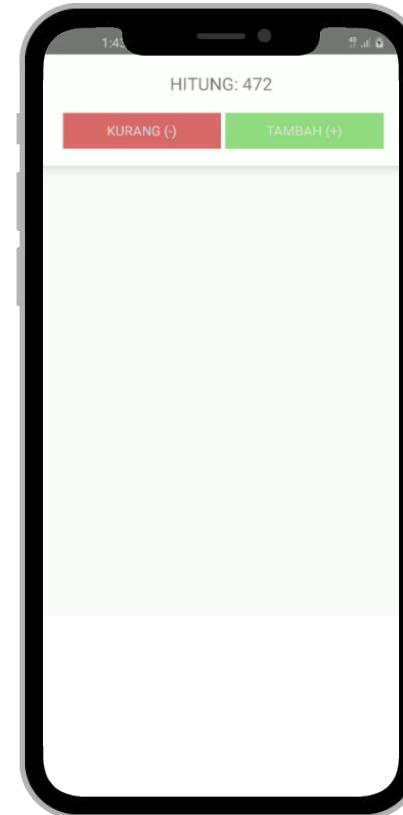
export default angkaReducer;
```



Data pada properti **angkaSekarang** yang terdapat pada Reducer telah tampil di komponen Counter.

Caranya dengan menampilkan variabel **this.props.reducerAngka.angkaSekarang**.





Perlu diingat!

Pemanggilan 'this.props.reducerAngka.**angkaSekarang**' perlu disesuaikan dengan properti yang kamu masukan pada object initialState yang terdapat pada Reducer ya.

Jadi kalau di Reducer kamu tulis propertinya adalah **angkaHitung**, maka ketika pemanggilannya di komponen menjadi 'this.props.angkaSekarang.**angkaHitung**'.



7. Trigger Action

Nah selanjutnya kamu akan mencoba trigger action ketika user mengklik tombol "Tambah +".

Caranya adalah buat sebuah function dengan nama **handleTambahButton**, function ini nantinya akan ke trigger ketika user mengklik tombol "Tambah".

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

import {connect} from 'react-redux';
import {tambahinAngka, kuranginAngka} from './actions/index';

class Counter extends React.Component {

  handleTambahButton = () =>{
    this.props.tambahinAngka()
  }

  render() {
    return (
      <View style={styles.counterContainer}>
        <Text style={styles.counterInfo}>
          HITUNG: {this.props.reducerAngka.angkaSekarang}
        </Text>
        <View style={styles.counterBtnContainer}>
          <TouchableOpacity
            style={styles.counterButtonMinus}>
            <Text style={styles.counterText}>KURANG (-)</Text>
          </TouchableOpacity>
          <TouchableOpacity
            style={this.handleTambahButton}
            style={styles.counterButtonPlus}>
            <Text style={styles.counterText}>TAMBAH (+)</Text>
          </TouchableOpacity>
        </View>
      </View>
    );
  }
}

const mapStateToProps = state => {
  return {
    reducerAngka: state.reducerAngka,
  };
};

const mapDispatchToProps = {
  tambahinAngka,
  kuranginAngka,
};

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



Misalnya kamu ingin ketika user menekan tombol "Tambah", maka nilai **angkaSekarang** yang berada di dalam reducer bertambah

1 angka.

Untuk mentrigger perubahan pada reducer tersebut, kamu perlu panggil action **tambahinAngka** yang telah terhubung dengan komponen Counter dengan cara
this.props.tambahinAngka()

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

import {connect} from 'react-redux';
import {tambahinAngka, kuranginAngka} from './actions/index';

class Counter extends React.Component {

  handleTambahButton = () =>{
    this.props.tambahinAngka()
  }

  render() {
    return (
      <View style={styles.counterContainer}>
        <Text style={styles.counterInfo}>
          HITUNG: {this.props.reducerAngka.angkaSekarang}
        </Text>
        <View style={styles.counterBtnContainer}>
          <TouchableOpacity
            style={styles.counterButtonMinus}>
            <Text style={styles.counterText}>KURANG (-)</Text>
          </TouchableOpacity>
          <TouchableOpacity
            style={this.handleTambahButton}
            style={styles.counterButtonPlus}>
            <Text style={styles.counterText}>TAMBAH (+)</Text>
          </TouchableOpacity>
        </View>
      </View>
    );
  }
}

const mapStateToProps = state => {
  return {
    reducerAngka: state.reducerAngka,
  };
};

const mapDispatchToProps = {
  tambahinAngka,
  kuranginAngka,
};

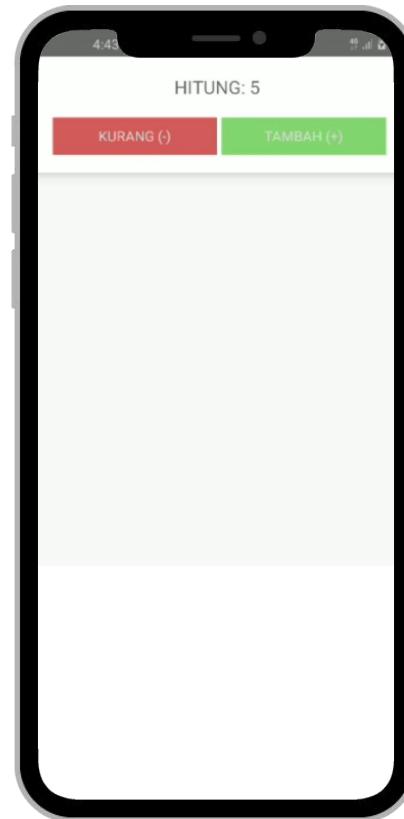
export default connect(mapStateToProps, mapDispatchToProps)(Counter);
```



Ini dia hasilnya~

Sekarang saat user menekan tombol "Tambah", maka action tambahinAngka akan ter-trigger dan menyebabkan data yang disimpan pada Reducer berubah.

Karena component Counter telah menampilkan data yang bersumber dari reducer, maka setiap ada perubahan pada reducer, data pada UI akan tetap berubah.





Nah, sekarang kamu sudah berhasil deh menghubungkan data yang berada pada redux store/reducer ke component UI!





**Biar level skill-mu naik, kamu harus coba
tantangan berikut ini nih!**



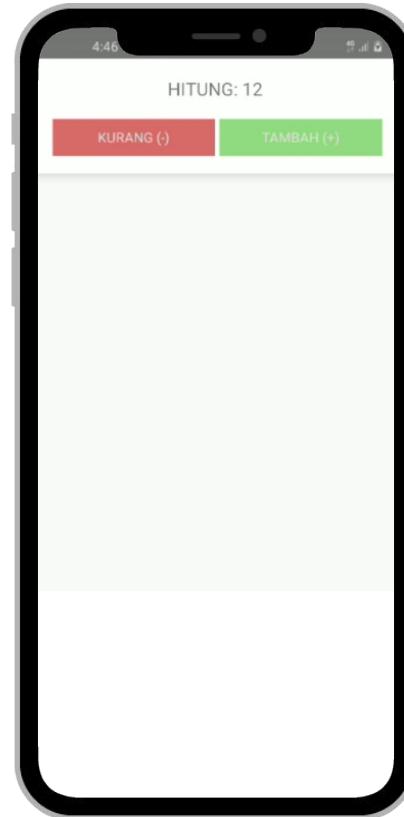


Ini dia tantangannya!

Buatlah aplikasi counter ini berfungsi ketika user mengklik button "Kurang". Jadi kalau nilai hitung saat ini adalah 12, maka ketika setiap kali user mengklik tombol "Kurang", nilainya akan berkurang 1.

Lakukan step-by-step nya sama persis seperti yang telah kita lakukan pada Tombol "Tambah" tadi ya!

Semangattt, guys! 💪





Oke, next!

Bayangin project aplikasimu sudah semakin rumit, lalu kamu harus cek apakah ada bug atau nggak?

Tanpa bantuan Doraemon pasti kamu bakal kesulitan, guys!





Tapi tenang aja~ Walaupun nggak ada Doraemon, tapi ada **Redux Logger**

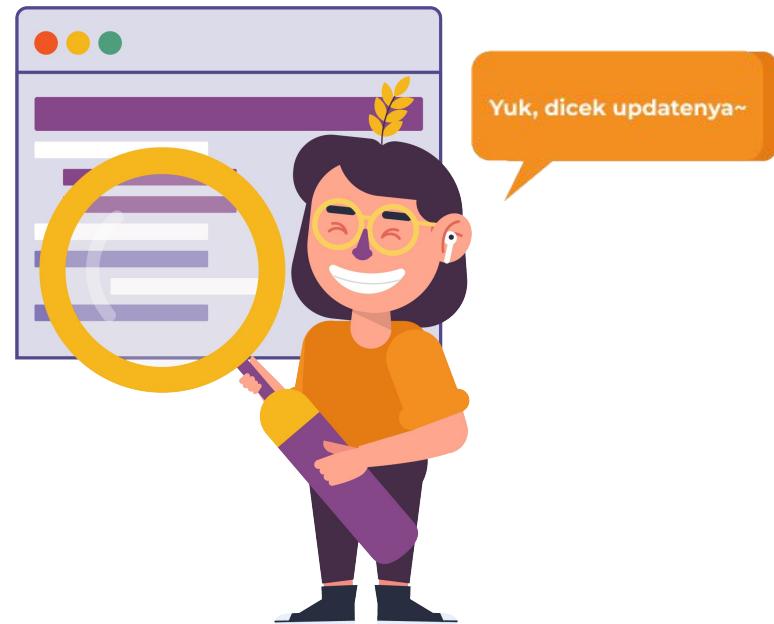
Yuk langsung aja kita kepoin apa sih **Redux Logger** ini!





Apa itu Redux Logger?

Redux Logger adalah sebuah **tools yang dapat membantu melacak perubahan** yang terjadi ketika suatu action yang menyebabkan perubahan pada Reducer dijalankan.

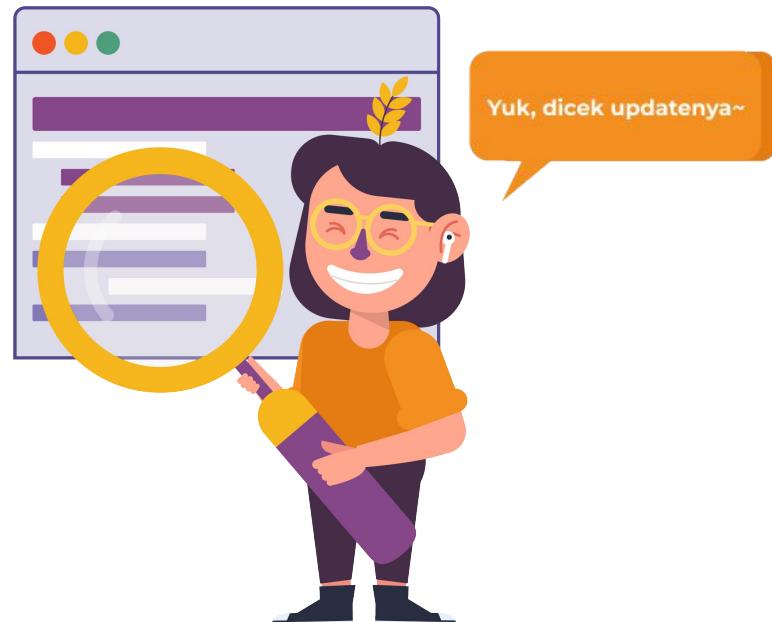




Kenapa kita perlu Redux Logger?

Well, sebenarnya Redux Logger bukan suatu hal yang wajib kamu pakai sih, tapi Redux Logger bisa membantumu jadi lebih mudah melakukan debugging (mengidentifikasi bug) saat aplikasimu semakin kompleks atau semakin banyak fiturnya.

Prinsipnya, kalau ada yang gampang, kenapa harus yang susah, kan? Hehe..





Kalau merasa perlu, yuk install dulu Redux Logger ini

Cara menginstall Redux Logger ke project react native gampang dan sederhana. Kamu hanya perlu mengetik command:

Npm install redux-logger





Lakukan ini setelah instalasi berhasil!

Selanjutnya, **import library redux-logger pada file index.js store** yang terdapat pada folder src/store.

```
// src/store/index.js

import {createStore, combineReducers} from 'redux';
import reduxLogger from 'redux-logger';

// import reducer
import angkaReducer from '../reducers/index';

const allReducers = combineReducers({angkaReducer: angkaReducer});

export const store = createStore(allReducers);
```



Masih lanjut~

Kemudian, **import method applyMiddleware dari library redux.** Method applyMiddleware ini berfungsi sebagai jembatan apabila kita ingin menambahkan middleware redux pada project kita.

```
// src/store/index.js

import {applyMiddleware, createStore, combineReducers} from 'redux';
import reduxLogger from 'redux-logger';

// import reducer
import angkaReducer from '../reducers/index';

const allReducers = combineReducers({angkaReducer: angkaReducer});

export const store = createStore(allReducers);
```



Selanjutnya, pada method applyMiddleware
masukan reduxLogger yang telah kita import sebagai parameter.

```
// src/store/index.js

import {applyMiddleware, createStore, combineReducers} from 'redux';
import reduxLogger from 'redux-logger';

// import reducer
import angkaReducer from '../reducers/index';

const allReducers = combineReducers({angkaReducer: angkaReducer});

export const store = createStore(allReducers, applyMiddleware(reduxLogger));
```



Kalau Redux Logger sudah berhasil diinstal dan bisa digunakan, kita lanjut ke cara menampilkan mode debugging yuk~





Mari kita mulai!

Untuk menampilkan mode debugging, kamu harus klik cursor mouse terlebih dahulu ke terminal metro bundler.

Klik terminal/cmd metro bundler tersebut kemudian **tekan huruf "D"** pada keyboard.

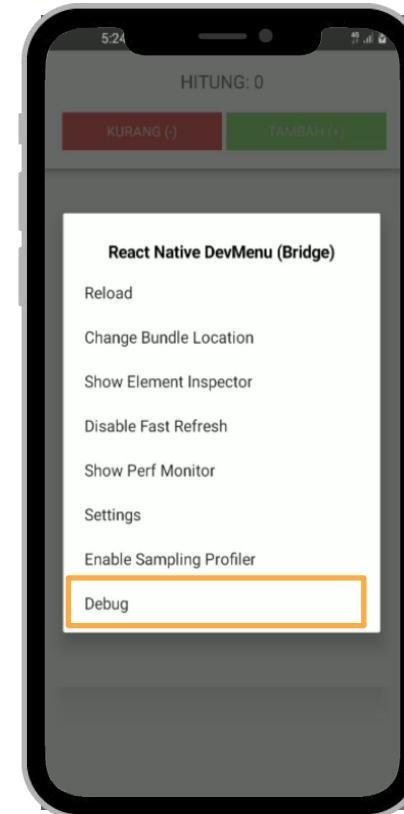
```
aldi — Metro — node -l launchPackager.command — 80x24
info Reloading app...
BUNDLE ./index.js

LOG Running "mytodoapp" with {"rootTag":191}
LOG Running "mytodoapp" with {"rootTag":201}
```



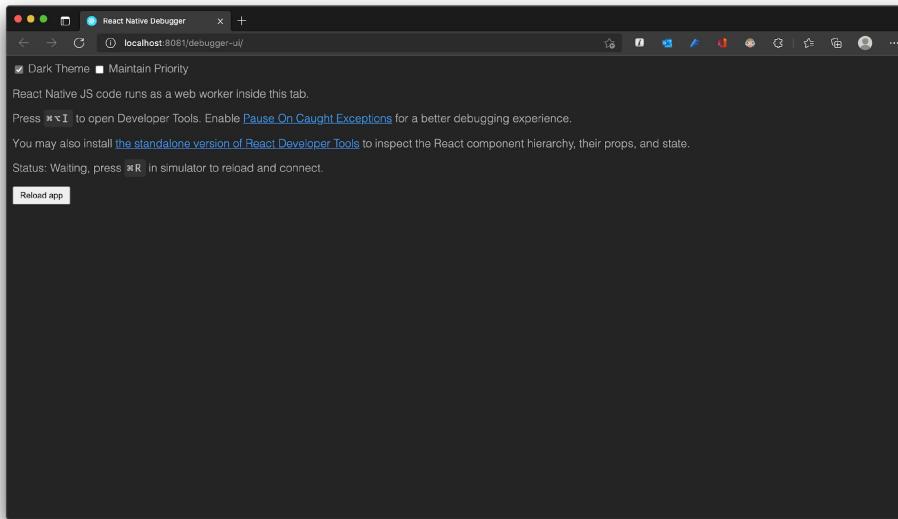
Setelah itu akan muncul tampilan Development Menu seperti gambar di samping pada aplikasi react native kamu.

Kemudian pada pop up menu tersebut **pilih "Debug"** yang terletak di paling bawah.



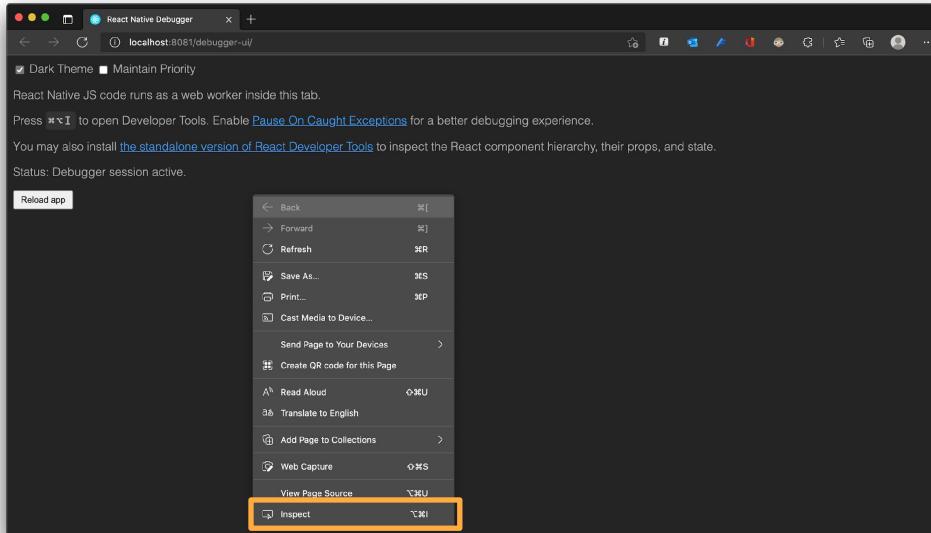


Setelah menekan tombol "Debug", maka secara otomatis komputermu akan **membuka sebuah browser dan akan menampilkan React Native Debugger** seperti tampilan di bawah ini.



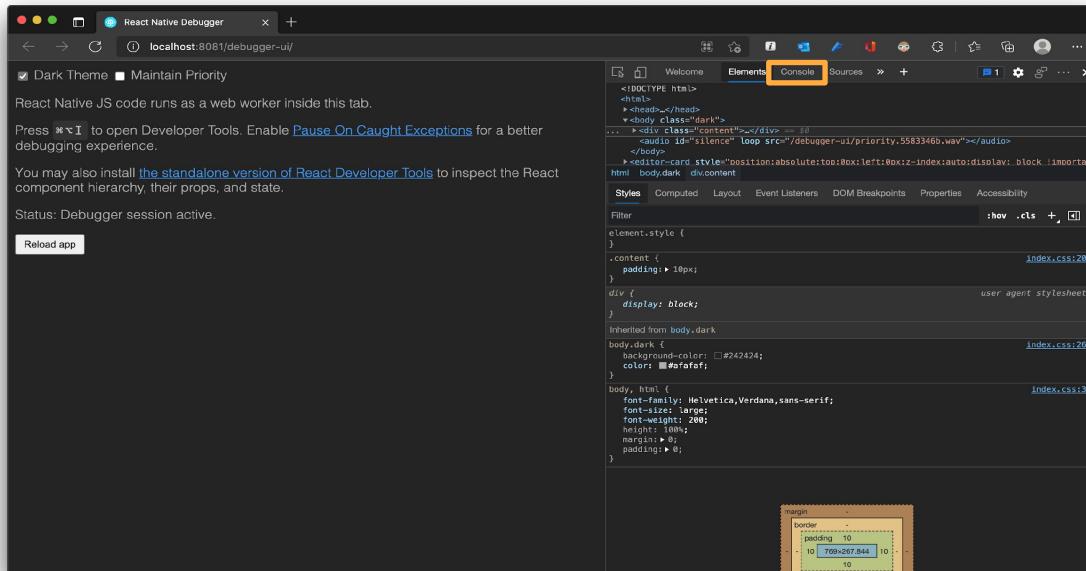


Klik kanan pada browser, dan **pilih menu "Inspect"**



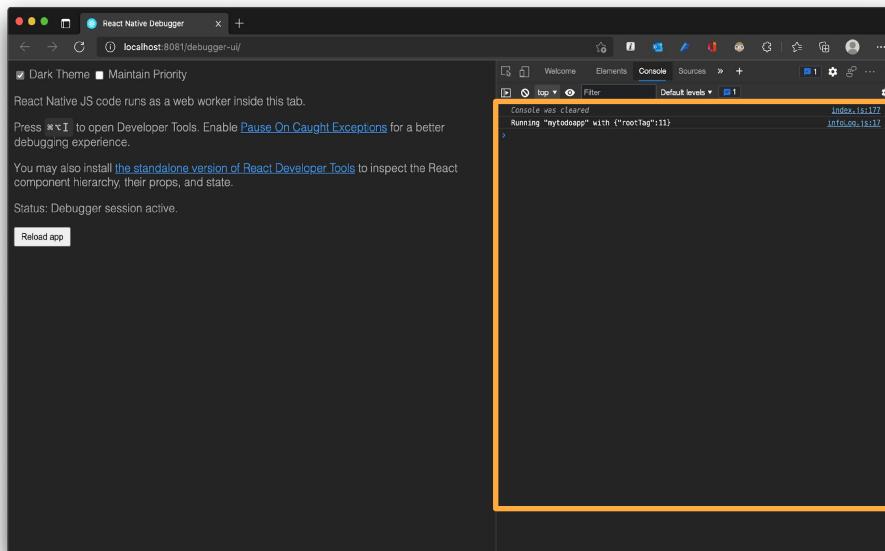


Beberapa menu pada tab akan muncul, **klik tab "Console"**





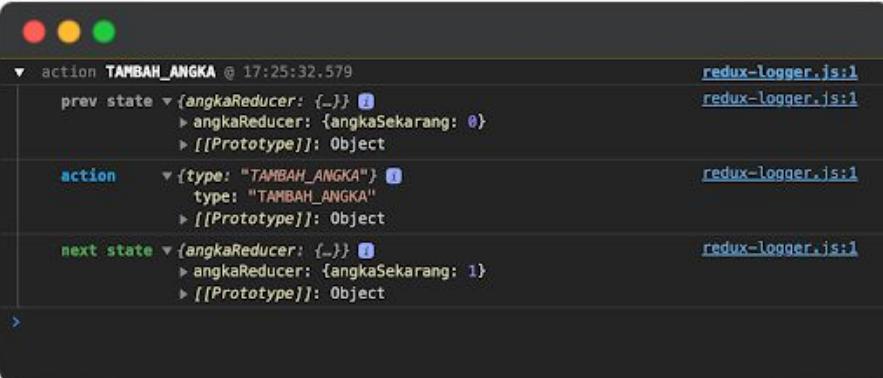
Pada bagian tab "Console" di browser ini, kamu bisa melihat langsung seluruh output dari console.log yang kamu tulis pada project React Native-mu.





Ketika kamu mentrigger sebuah action, setiap action yang kamu trigger akan ditampilkan oleh Redux-Logger pada tab "Console", Sehingga kamu bisa melakukan tracking terhadap setiap perubahan reducer yang disebabkan oleh action.

Ini akan sangat membantu ketika aplikasimu sudah berkembang menjadi lebih kompleks, dan Reducer yang kamu gunakan juga semakin banyak



The screenshot shows a browser's developer tools console with a dark theme. It displays the output of a Redux logger. The log shows an action being triggered, along with the previous state, the action object itself, and the resulting next state. The code snippets shown are from 'redux-logger.js:1'.

```
▼ action TAMBAH_ANGKA @ 17:25:32.579
  prev state ▼ {angkaReducer: {...}}
    ▶ angkaReducer: {angkaSekarang: 0}
    ▶ [[Prototype]]: Object
  action   ▷ {type: "TAMBAH_ANGKA"} ⓘ
    type: "TAMBAH_ANGKA"
    ▶ [[Prototype]]: Object
  next state ▼ {angkaReducer: {...}}
    ▶ angkaReducer: {angkaSekarang: 1}
    ▶ [[Prototype]]: Object
```



Nah, sekarang kamu sudah tahu cara nampilin data dari Redux Store/Reducer ke dalam UI dan paham cara pakai Redux Logger untuk bantu proses debugging aplikasimu dehh..

Congratulations!!



Saatnya kita Quiz!





1. Andi ingin membuat sebuah aplikasi menggunakan redux. Dan Andi ingin membuat dua jenis reducer, reducer untuk menyimpan data profile user, dan reducer untuk menyimpan data list berita. Method manakan yang harus digunakan Andi?

- A. createMiddleware()
- B. configureStore()
- C. combineReducers()



1. Andi ingin membuat sebuah aplikasi menggunakan redux. Dan Andi ingin membuat dua jenis reducer, reducer untuk menyimpan data profile user, dan reducer untuk menyimpan data list berita. Method manakan yang harus digunakan Andi?

- A. createMiddleware()
- B. configureStore()
- C. **combineReducer()**

Pada case ini andi ingin membuat dua reducer, yaitu `profileReducer` dan `listBeritaReuder`, andi dapat menggunakan method `combineReducer` untuk menggabungkan beberapa reducer menjadi satu.



2. Berapa argument yang bisa dimasukan pada method createStore() pada Redux?

- A. 1
- B. 2
- C. 3



2. Berapa argument yang bisa dimasukan pada method createStore() pada Redux?

- A. 1
- B. 2
- C. 3

Method createStore() bisa menerima 3 argument yaitu createStore(reducer,[preloadedState],[enhancer]), argument pertama sifatnya mandatory, dan sisanya optional

Lebih lanjut dapat dibaca di

<https://redux.js.org/api/createstore>



3. Reducer pada dasarnya adalah hanya function biasa. Apakah menurut kamu pernyataan diatas benar?

- A. Benar
- B. Tidak Benar
- C. Benar dan tidak benar tergantung kebutuhan reduxnya



3. Reducer pada dasarnya adalah hanya function biasa. Apakah menurut kamu pernyataan diatas benar?

- A. Benar
- B. Tidak Benar
- C. Benar dan tidak benar tergantung kebutuhan reduxnya

Reducer pada dasarnya hanya function biasa pada javascript yang mereturn data



4. Dibawah ini manakah library atau tools yang bertujuan untuk membantu proses debugging pada Redux?

- A. Redux-Devtools
- B. Redux-Logger
- C. Dua-duanya benar



4. Dibawah ini manakah library atau tools yang bertujuan untuk membantu proses debugging pada Redux?

- A. Redux-Devtools
- B. Redux-Logger
- C. **Dua-duanya benar**

Redux-Devtools dan Redux-logger adalah tools dan library untuk mempermudah proses debugging



5. Dibawah ini manakah yang merupakan helper method yang terdapat pada react-redux?

- A. view()
- B. assist()
- C. connect()



5. Dibawah ini manakah yang merupakan helper method yang terdapat pada react-redux?

- A. view()
- B. assist()
- C. connect()

connect() adalah helper method pada react-redux yang bertujuan untuk menghubungkan komponen dengan redux



6. Menurut kamu apakah library redux dan react-redux itu sama?

- A. Iya sama
- B. Tidak sama
- C. Keduanya benar



6. Menurut kamu apakah library redux dan react-redux itu sama?

- A. Iya sama
- B. **Tidak sama**
- C. Keduanya benar

Redux adalah package yang bukan khusus react saja, pada dasarnya redux juga dapat dipake di framework javascript lain seperti Angular, Vue.js dll. React-redux adalah library yang membantu mengkoneksikan redux pada React.js



7. Komponen manakah yang terdapat pada react-redux yang bertujuan sebagai pembungkus semua komponen pada aplikasi react?

- A. <Container >
- B. <Actions >
- C. <Provider >



7. Komponen manakah yang terdapat pada react-redux yang bertujuan sebagai pembungkus semua komponen pada aplikasi react?

- A. <Container >
- B. <Actions >
- C. <Provider >

Provider adalah komponen yang di gunakan pada Root file Component yang bertujuan untuk membungkus seluruh komponen yang ada pada aplikasi react tersebut



8. Dibawah ini yang merupakan method pada object store yang berfungsi mendapatkan semua data pada store adalah

- A. store.getState()
- B. store.dispatch()
- C. store.subscribe()



8. Dibawah ini yang merupakan method pada object store yang berfungsi mendapatkan semua data pada store adalah

- A. `store.getState()`
- B. `store.dispatch()`
- C. `store.subscribe()`

`store.getState` adalah method dari object store yang akan mereturn semua data yang terdapat pada store



9. Selain melalui props, kita juga dapat mentrigger action secara langsung melalui object store. Method mana yang digunakan untuk mentrigger action dengan object store?

- A. store.getState()
- B. store.dispatch()
- C. store.subscribe()



9. Selain melalui props, kita juga dapat mentrigger action secara langsung melalui object store. Method mana yang digunakan untuk mentrigger action dengan object store?

- A. store.getState()
- B. **store.dispatch()**
- C. store.subscribe()

Store.dispatch adalah sebuah method yang berasal dari object store yang berfungsi mentrigger sebuah action



10. Apa yang salah dengan dengan contoh action dibawah ini?

- A. Action harus berbentuk anonymous function
- B. Action harus return sebuah object dengan properti 'type'
- C. Action harus return sebuah object dengan properti 'Type' dan 'payload'



```
function addAngka(){
    return {
        typeAction: 'TAMBAH_ANGKA'
    }
}
```

10. Apa yang salah dengan dengan contoh action dibawah ini?

- A. Action harus berbentuk anonymous function
- B. **Action harus return sebuah object dengan properti 'type'**
- C. Action harus return sebuah object dengan properti 'Type' dan 'payload'



```
function addAngka(){
  return {
    typeAction: 'TAMBAH_ANGKA'
  }
}
```

Action pada dasarnya hanyalah sebuah function biasa yang mereturn object dengan properti 'type' didalamnya. Minimal dalam object yang di return harus terdapat properti 'type'. Jika tidak maka akan terjadi error.



11. Apa yang terjadi Kode komponen Terhubung Dengan redux pada gambar disamping dijalankan?

- A. Ketika komponen componentDidMount akan mentrigger sebuah action dengan nama tambahAngka,
- B. Akan muncul error
- C. Tidak terjadi apa-apa

```
•••  
  
import React from 'react';
import {Text} from 'react-native';
import {connect} from 'react-redux';
import {tambahAngka} from '../action/movies.js'  
  
class Counter extends React.Component {
  componentDidMount(){
    tambahAngka()
  }
  render(){
    return(
      <Text>
        {this.props.curentAngka}
      </Text>
    )
  }
}  
  
const mapStateToProps = (state) =>{
  return {
    ... state
    curentAngka: state.counter.currentNumber
  }
}
const mapDispatchToProps = {
  tambahAngka
}
export default connect(mapStateToProps, mapDispatchToProps)(Counter)
```



11. Apa yang terjadi Kode komponen Terhubung Dengan redux pada gambar disamping dijalankan?

- A. Ketika komponen componentDidMount akan mentrigger sebuah action dengan nama tambahAngka,
- B. Akan muncul error
- C. **Tidak terjadi apa-apa**

Tidak terjadi apa-apa karena pada componentDidMount yang dijalankan adalah sebuah function biasa, bukan sebuah action, function action dapat dijalankan pada variabel this.props

```
•••  
  
import React from 'react';
import {Text} from 'react-native';
import {connect} from 'react-redux';
import {tambahAngka} from '../action/movies.js'  
  
class Counter extends React.Component {
  componentDidMount(){
    tambahAngka()
  }
  render(){
    return(
      <Text>
        {this.props.curentAngka}
      </Text>
    )
  }
}  
  
const mapStateToProps = (state) =>{
  return {
    ... state
    curentAngka: state.counter.currentNumber
  }
}
const mapDispatchToProps = {
  tambahAngka
}
export default connect(mapStateToProps, mapDispatchToProps)(Counter)
```

Terima Kasih!



Next Topic

loading...