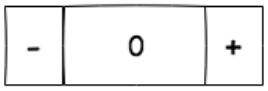
Test Soal React IRVAN

1. Jelaskan yang dimaksud dengan State dan Props?

   States data *private* sebuah component. Data ini hanya tersedia untuk component tersebut dan tidak bisa di akses dari component lain. Component dapat merubah statenya sendiri.

   Contoh :



Cara kerja component ini sederhana :

1. default valuenya adalah 0.
2. ada tombol minus dan plus untuk mengubah value nya.

Code :

```
class Counter extends React.Component{
  // #1 inisialisasi state
  state={
    value :0
  }

  // #2 method untuk merubah state
  minus=()=>{
    let currentValue=this.state.value
    this.setState({ value : currentValue - 1 })
  }

  // #2 method untuk merubah state
  plus=()=>{
    let currentValue=this.state.value
    this.setState({ value : currentValue + 1 })
  }

  render(){
    // #3 read component state
    let currentValue=this.state.value

    // #4 call plus method
    this.plus()
    console.log(currentValue) //=> 1

    // #4 call minus method
    this.minus()
    console.log(currentValue) //=> -1

  }
}
```

Props dari *Property*. Ini cukup simple, apalagi jika anda sudah terbiasa dengan HTML, ini mirip seperti atrribute pada tag HTML. Dalam pembuatannya, jika dalam *functional component* maka *prop* ini adalah parameternya. Jika componentnya dalam bentuk class maka prop ini property dari class nya yang di akses melalui keyword '*this'*. **Props** itu *read-only* dan umumnya digunakan untuk komunikasi data component dari *parent component* ke *child componen*t.

Contoh:

Untuk mengakses object component **props**, kita menggunakan sebuah ekspresi yang di sebut *this.props*.

Code :

```
render:function(){
      console.log("Props object comin up!");
      console.log(this.props);
      console.log("That was my props object");
      return <h1>Hello world</h1>;
}
```

2. Jelaskan yang dimaksud dengan "*Component DID Mount, Component Did Update, Component Will Unmount"* !

   Component DID Mount salah satu fase dari *React component lifecycle. Component DID Mount* adalah fungsi yang dieksekusi setelah *render* pertama kali digunakan. Pada *lifecycle* inilah http request/ajax dilakukan karena *lifecycle* ini component telah masuk ke DOM/*already mounted*.
   Contohnya :
   - ❖ Permintaan jaringan

   Component DID Update merupakan salah satu fase dari *React component lifecycle. Component DID Update* dipanggil setelah function *render*. Sama dengan Component DID Mount, tuntion ini dapa digunakan untuk melakukan operasi DOM setelah data diperbaharui.

   Component Will Unmount merupakan salah satu fase dari *React component lifecycle*. *React component lifecycle* adalah tahap berurutan dari kondisi komponen ketika digunakan. *Component Will Unmount* adalah satu-satunya method yang akan di eksekusi pada proses unmounting atau method yang akan dieksekusi sebelum *component* dihapus atau dihilangkan dari DOM. Fungsi ini berguna ketika aksi-aksi yang berhubungan dengan pembersihan diperlukan. Membuang timer,
   - ❖ Membatalkan permintaan jaringan,
   - ❖ Membersihkan any Subscriptions yang dibuat di componentDidMount.

3. Jelaskan cara mengambil data dan mengirim data dalam React Component(get dan post)!
   Cara mengambil data dalam React Component
   Cara pengambilan data (fetch) dalam React ada 3 cara yaitu **Normal Fetch, Asych/await Fecth dan Export/Import Your Ways**

   ### 1. Normal Fetch

```
import React, { Component } from 'react'

class App extends Component {
    state = {
      data: []
    }

    componentDidMount(){
      const urlFetch = fetch('https://yourweb/api/endpoint')

      urlFetch.then(res => {
        if( res.status === 200)
          return res.json()
      }).then( resJson => {
        this.setState({
          data: resJson
        })
      })

    }

    render(){
      return (<YourComponent>)
    }
}
```

Pada **Class App** (default **CRA**) saya mendefinisikan sebuah **State** dengan nama "**data**" yang diberi nilai **Array** [].

```
state = {
    data : []
}
```

Kemudian,
menambahkan *lifecycle componentDidMount* yang dimana setiap fungsi atau perintah yang di tuliskan pada *method* atau *function* tersebut dijalankan setelah *Component* telah selesai melakukan *render* **DOM**.

```
componentDidMount(){
    const urlFetch = fetch('alamaturl')
   urlFetch.then( res => {
      if(res.status === 200)
         return res.json()
   }).then( resJson => {
      this.setState({
          data: resJson
      })
   })
}
```

- **const urlFetch :** Definisi konstanta ***urlFetch*** yang di beri nilai ***Fetch***berserta parameter berupa String untuk alamat url yang akan kita Request data. Kembalian nilai dari fungsi ***Fetch*** yakni berupa ***Object Promise***, ***Object Promise*** memiliki 3 *State* yakni ***Pending***, ***Fulfilled***, & ***Reject***. keadaan pertama yakni ***Pending***, didalam keadaan ini akan berubah menjadi salah satu dari ***Fulfilled*** atau ***Reject***, ***Fulfilled*** ketika Permintaan ***Fetch*** berhasil dan berjalan tanpa masalah, ***Reject*** ketika Permintaan gagal (Penyebab yang umum yakni ***CORS***). ***Finally,*** nilai yang di kembalikan yakni berupa ***promise*** juga, yang miliki 2 ***method promise.then*** dan ***promise.catch.***
- urlFetch.then : Pada method then, Menge-check apakah status request dengan nilai 200 , jika benar maka mengembalikan nilai berupa res.json() kemudian memanggil kembali method then untuk mengambil nilai kembalian yang telah di dapatkan dan mengatur kembali state.datadengan fungsi this.setState dengan nilai JSON (resJson).

2. Async/Await Fecth

```
import React, { Component } from 'react'

class App extends Component {
    state = {
      data: []
    }

    componentDidMount(){
      const urlFetch = fetch('https://yourweb/api/endpoint')

      urlFetch.then(res => {
        if( res.status === 200)
          return res.json()
      }).then( resJson => {
        this.setState({
          data: resJson
        })
      })

    }

    render(){
      return (<YourComponent>)
    }
}
```

Cara ke-2 dengan memanfaatkan ***Async/Await***, Pada cara ini sebaiknya memahami ***Promise*** yang telah di tuliskan pada cara pertama agar dapat memahami bagaimana proses kerja ***Async Function*** dan ***Await***.
Async Function : Membuat ***Object asynchronous Function*** , Dengan menambahkan kata ***async*** saat mendefinisikan sebuah ***function***.

- *Await* : Untuk menunda ***statement*** atau pengerjaan di sebuah ***Async Function*** sampai ***object Promise*** selesai. (***Await Promise***)

```
const FungsiAsync = async () => {
   const urlFetch = await fetch('url'} # fetch is Promise object
   # fetch kembaliannya dalam bentuk promise , maka
   # baris code setelah await fetch akan di kerjakan sampai promise
   # atau fungsi fetch tersebut selesai.
   # jika telah selesai , maka lanjut pada statement return dibawah
   # ini.
   # return jika 'json' in urlFetch benar maka
   # return await urlFetch.json() , jika false maka
   # return Array []
   return 'json' in urlFetch ? await urlFetch.json() : []
   # mengapa await urlFetch.json() ?
   # urlFetch.json masih bernilai Object Promise , dengan
   # menambahkan await maka kita akan resolve dan
   # mendapatkan nilai JSON
```

3. Export/Import Your Ways

Pertama kali, Buat sebuah ***File*** , Kita beri nama "**AsyncFetch.js**" pada folder ***src***, Berikut code pada file tersebut.

```
const AsyncFetch = async (url) => {
    const urlFetch = await fetch(url)
    return urlFetch.status === 200 && 'json' in urlFetch ?
```

```
        await urlFetch.json() : []
    }
    export default AsyncFetch
```

- ***const AsyncFetch :*** fungsi yang digunakan dan di jelaskan pada cara ke 2 diatas.
- ***export default AsyncFetch:*** degan menambahkan Keyword export default maka fungsi tersebut secara langsung digunakan ketika component atau module tersebut di import di component atau module lainnya.

Kemudian **save** dan kembali ke **App.js** , untuk menggunakan AsyncFetch.js,

- ***Import*** dulu ***module AsyncFetch.***
- Pada ***async component Did Mount***, langsung saja ***setStateAsync*** dengan ***key*** data pada ***state*** di beri nilai **await AsyncFetch(url)** dan mendapatkan nilai ***JSON*** dari ***URL*** yang di **Request.**

```
import AsyncFetch from './AsyncFetch'

  async componentDidMount(){
    const url = "your url"
    this.setStateAsync({
        data: await AsyncFetch(url)
    })
  }
```

4. Rancanglah sebuah program yang mempassing data dari Component Parent ke Component Children! Sertakan Contoh Codenya!
   Untuk mempassing data dari Parent ke children biasa menggunakan metode **Props**

```
class StatefulComponent extends React.Component {
        state={
                items=[]
        }

        _actionDelete=(ID) ➔ {
        }

        Render () {
                let todos = this.props.todos
                return (
                <View>
                        <TodoItems
                                todos={this.state.todos}
                                actionDelete={this._actionDelete}
                        />
                </View>
                )
        }
}
```

- Dari contoh aplikasi di atas, *Parent Component <App/>* mengirim data ke*child component <TodoItems* ***todos****={data}/>* dengan *props*. Selanjutnya data tersebut bisa dibaca di component *TodoItems* dengan perintah *this.props.****todos***
- Data dalam props bisa berupa *object* atau *function*. Pada contoh ini method *_actionDelete()* di kirim melalui props yang nantinya akan menjadi *callback* dan dieksekusi di component *TodoItems*