



# Komponen React

**Silver**- Chapter 2 - Topic 3

---

**Selamat datang di Chapter 2 Topic 3 pada course  
React Native dari Binar Academy!**





# Haalloo teman-teman! 🙋

Kita lanjut ke **Chapter 2 Topic 3 React Native**. Di Topic 2 kemarin kita sudah belajar tentang HTML dan CSS. Di sesi kali ini kita bakal belajar tentang **Komponen pada React** dan hal-hal yang terkait dengan itu.

Cuss, langsung saja kita mulai!



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

- Konsep dan cara kerja Komponen React
- Pembuatan Komponen React
- Konsep dan cara kerja Props dan State
- Konsep dan cara kerja Components Lifecycle pada React

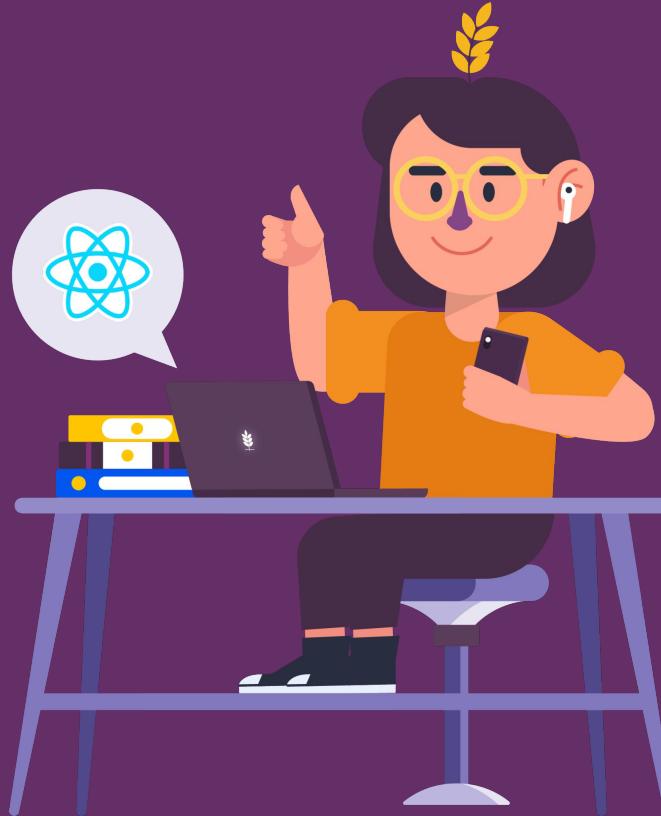




## Tahu nggak sih?

**React.JS** berfungsi mengatur layer pada tampilan desktop maupun aplikasi mobile.

Dengan React, kamu bisa bikin jenis komponen yang bisa kamu pakai lagi, tanpa harus membangunnya dari awal.



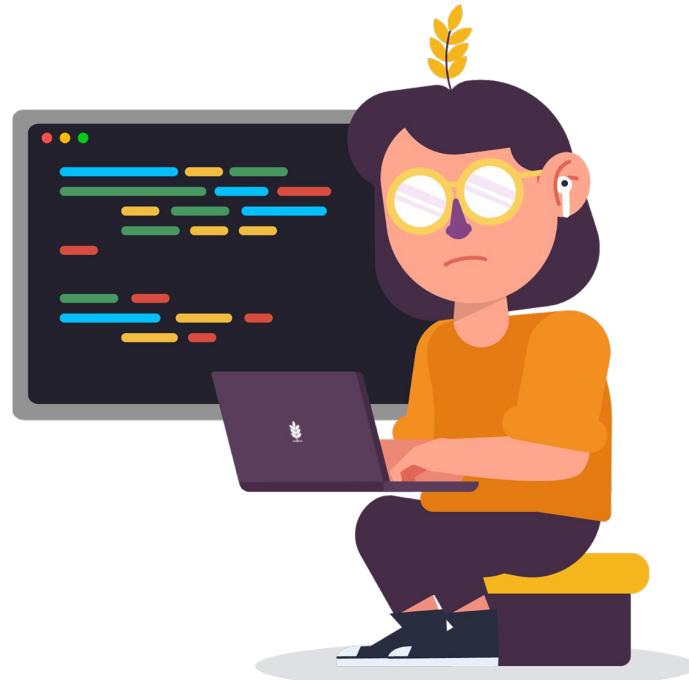


React adalah **JavaScript library untuk membuat UI (User Interface)** pada Web dan Mobile Application.

React dikembangkan oleh Facebook sebagai salah satu library Open Source pada 2013 dan dipakai pada produk aplikasi mereka seperti Facebook, WhatsApp, Instagram, dan lain-lain.

React memungkinkan developer untuk bikin aplikasi web yang bisa mengubah data, tanpa memuat ulang lamannya.

Tujuan utamanya sih biar proses lebih cepat, terukur, dan sederhana.





Jadi hari ini kita akan lakukan beberapa hal berikut ini dulu:

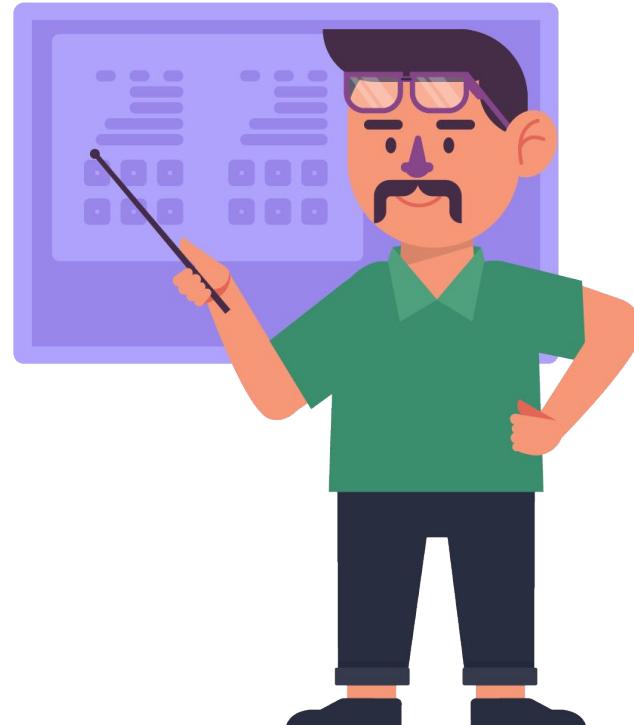
- Setup project untuk React.JS dengan create-react-app
- Memodifikasi komponen aplikasi React.JS
- Melakukan styling untuk aplikasi React.JS

Untuk melakukan hal di atas, pastikan komputermu sudah install aplikasi berikut:

- Runtime: Node.JS
- Package: npm / yarn

## Catatan

**Proses setup bisa memakan waktu. Pastikan juga koneksi internetmu stabil dan sebaiknya matikan dahulu aplikasi lain yang nggak dipakai.**

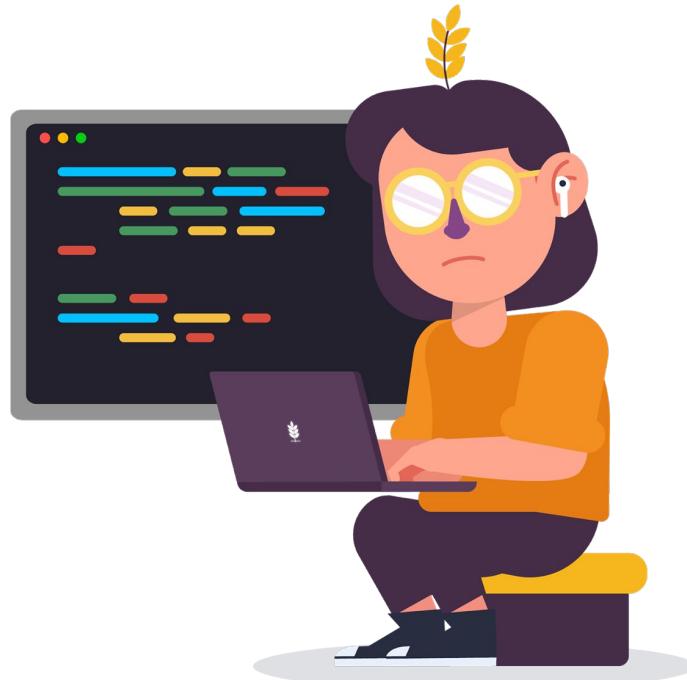




## Pertama, Setup Project!

Jangan lupa **manfaatkan package create-react-app yang merupakan template untuk bikin project React.JS.**

Developer pemula sangat disarankan pakai metode ini lebih dulu. Kalau sudah paham proses kerja React.JS, baru bisa mengembangkan struktur project-nya sendiri.





## Nih, tiga langkah setup project dengan create-react-app:

1. Install package tersebut via terminal dengan mengetikkan command:

```
$ npm install -g create-react-app
```

1. Buat project baru via terminal. Misalnya kamu mau nama project-nya adalah belajar-react-binar, maka command di terminal sebagai berikut:

```
$ create-react-app belajar-react-binar
```

1. Setelah proses download selesai, akan ada folder baru dengan nama belajar-react-binar. Buka folder tersebut dengan visual studio code.

```
PS C:\Users\Aldi> npm i -g create-react-app  
[██████████] / fetchMetadata: sill re
```

```
PS C:\Users\Aldi> create-react-app belajar-react-binar  
Creating a new React app in C:\Users\Aldi\belajar-react-binar.
```



```
▶ node_modules
└─┬ public
  └── favicon.ico
  └── index.html
  └── manifest.json
└─┬ src
  └── App.css
  └── App.js
  └── App.test.js
  └── index.css
  └── index.js
  └── logo.svg
  └── registerServiceWorker.js
└── .gitignore
└── package-lock.json
└── package.json
└── README.md
```

Lalu di mana file dan kode yang kamu buat tersimpan? Di folder structure, guys!

#### **node\_modules**

Folder yang berisi **paket-paket modul Nodejs**; semua library yang kamu install dengan npm akan tersimpan di sini.

#### **public**

Folder yang berisi **file untuk publik** seperti HTML, CSS, icon, dan gambar, dan aset publik lainnya

#### **src**

Folder yang berisi **kode dari aplikasi React.JS**, di sinilah kamu akan bikin komponen



**App.js**

File yang berisi **kode untuk component App atau component utama** dari aplikasi.



```
▶ node_modules
◀ public
  ★ favicon.ico
  ◁ index.html
  { } manifest.json
◀ src
  # App.css
  JS App.js
  JS App.test.js
  # index.css
  JS index.js
  📺 logo.svg
  JS registerServiceWorker.js
  ⚡ .gitignore
  { } package-lock.json
  { } package.json
  📄 README.md
```

### 📄 App.test.js

File yang berisi **kode untuk testing** komponen App

### 📄 index.js

File yang berisi **kode untuk render component App** ke Web Broser atau ke Mobile App.

### 📄 .gitignore

File yang berisi **list dari file atau folder** yang akan diabaikan oleh git

### 📄 package.json

File yang isinya JSON yang berisi **keterangan project dan daftar package-package eksternal** yang dibutuhkan.

Dalam React.JS ada yang disebut dengan JSX. File JSX memungkinkan penulisan kode HTML secara kombinasi.

Kayak apa bentuknya? Langsung saja lihat ke slide selanjutnya yuk.





## Apa itu JSX?

JSX (Javascript XML) adalah extension dari Javascript yang bikin kita bisa menulis kode HTML di dalam JavaScript. Sederhananya, **JSX adalah jenis file yang memperbolehkan kita untuk menulis kode HTML dicampur kode JavaScript.**

```
// File App.js

import React from 'react';

export default class App extends React.Component {
  render() {
    return (
      <div>
        <h1>INI JSX</h1>
        <p>HTML BERAMPUR JAVASCRIPT</p>
      </div>
    );
  }
}
```



Sama kayak XML dan HTML, JSX juga punya nama tag, atribut, dan elemen anak. Sebenarnya kamu bisa memakai React tanpa JSX, tapi nggak direkomendasikan karena lebih susah untuk dibaca dan ditulis.

### Komponen Tanpa JSX

```
1 class Btn extends React.Component {  
2     render() {  
3         let element =  
4             React.createElement('h1',  
5             {}, 'Ga pake JSX!');  
6         return element;  
7     }  
8 }
```

Tanpa JSX, kamu harus bikin elemen dengan method createElement().

### Komponen Dengan JSX

```
1 class Btn2 extends React.Component {  
2     render() {  
3         return (

# I Love JSX!

);  
4     }  
5 }  
6 }
```

Dengan JSX, kamu cuma perlu menulis sintaks XML atau HTML-nya saja.



Kok kedengarannya ribet ya?

Nggak ribet kok, asalkan kamu sudah memahami serba-serbi kode HTML.

FYI, beberapa aturan JSX juga perlu diterapkan supaya ngoding tetap nyaman dan compiler nggak error.

Apa saja aturannya? Cekidot!





## Peraturan pertama: Kode HTML harus nested

Nested di sini maksudnya **kode HTML harus punya awalan dan akhiran**.  
Nggak boleh ada yang ketinggalan supaya compiler nggak error.

```
// File App.js

import React from 'react';

export default class App extends React.Component {
  render() {
    return (
      <div>
        <h1>INI JSX</h1>
        <p>HTML BERCAMPUR JAVASCRIPT</p>
      </div>
    );
  }
}
```



```
// File App.js

import React from 'react';

export default class App extends React.Component {
  render() {
    return (
      <h1>INI JSX</h1>
      <p>HTML BERCAMPUR JAVASCRIPT</p>
    );
  }
}
```





## Peraturan kedua: Penulisan kode JavaScript di dalam HTML

Penulisan kode JavaScript di dalam tag HTML dibolehkan dengan menggunakan tag {}.

Ini berlaku juga untuk menulis comments.

```
// File App.js

import React from 'react';

export default class App extends React.Component {
  render() {
    const iniVariabelJS = 'meonnggggg'
    return (
      <div>
        <h1>{1+1}</h1>
        <h2>{iniVariabelJS}</h2>
        //End of the line Comment...
        /*Multi line comment...*/
      </div>
    );
  }
}
```



```
// File App.js

import React from 'react';

export default class App extends React.Component {
  render() {
    const iniVariabelJS = 'meonnggggg'
    return (
      <div>
        <h1>1+1</h1>
        <h2>iniVariabelJS</h2>
        //End of the line Comment...
        /*Multi line comment...*/
      </div>
    );
  }
}
```





Untuk menampilkan JSX ke browser, kamu membutuhkan rendering. Ada dua macam rendering yang bisa kamu pakai, yaitu **Rendering Dynamic Value** dan **Conditional Rendering**.

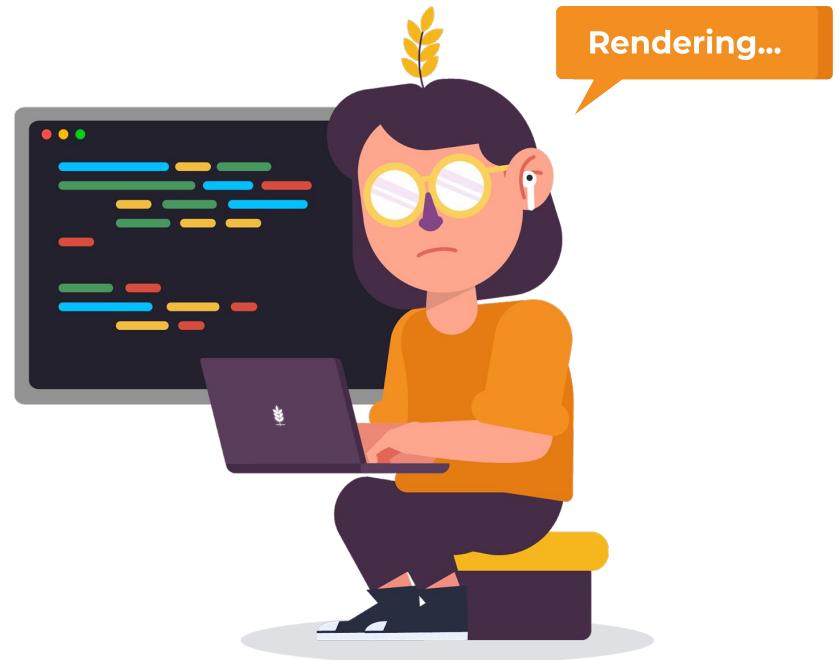




## Pertama, kita kenalan dulu sama Render Dynamic Value/Variabel

Rendering Dynamic Value artinya **menampilkan JSX yang bersifat dinamis** atau sering berubah-ubah.

Misalnya kamu punya variabel `totalProducts` yang value nya adalah 30000. Kamu bisa memanggil variabel `totalProducts` tersebut pada JSX, lalu JSX akan menampilkan 30000 di tampilan UI, karena 30000 adalah value dari variable `totalProducts`.

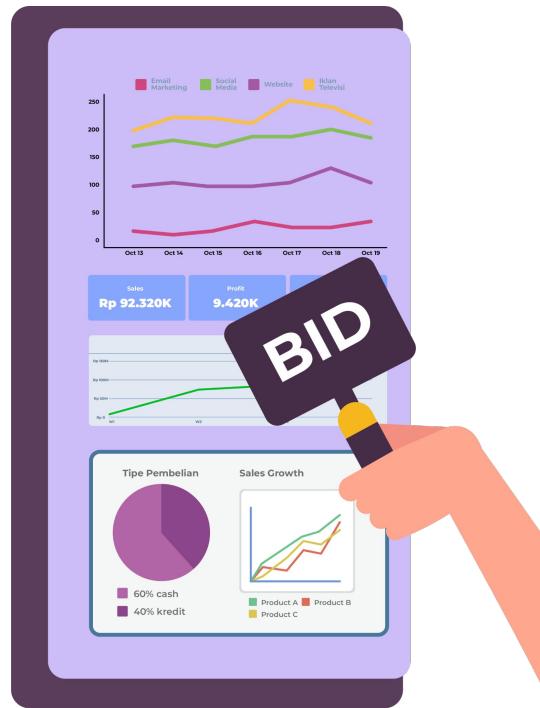




### Biar lebih jelas, kita pakai contoh saja ya~

Contoh data untuk Rendering Dynamic Value adalah saat kamu mau rendering data produk harga lelang.

**Data lelang kan berubah-ubah nilainya** tergantung penawaran, nah, **jadi tampilannya pun akan berubah-ubah** mengikuti perubahan data yang ada.





### Gak cuma permen, variabel juga perlu dibungkus, guys~

Di JSX, kamu bisa menampilkan value dari sebuah variabel ke UI. Untuk merender atau menampilkan sebuah variabel ke UI, kamu harus **bungkus variabel tersebut ke dalam tanda kurung kurawal {}**.



```
1 {namaVariabel}
```

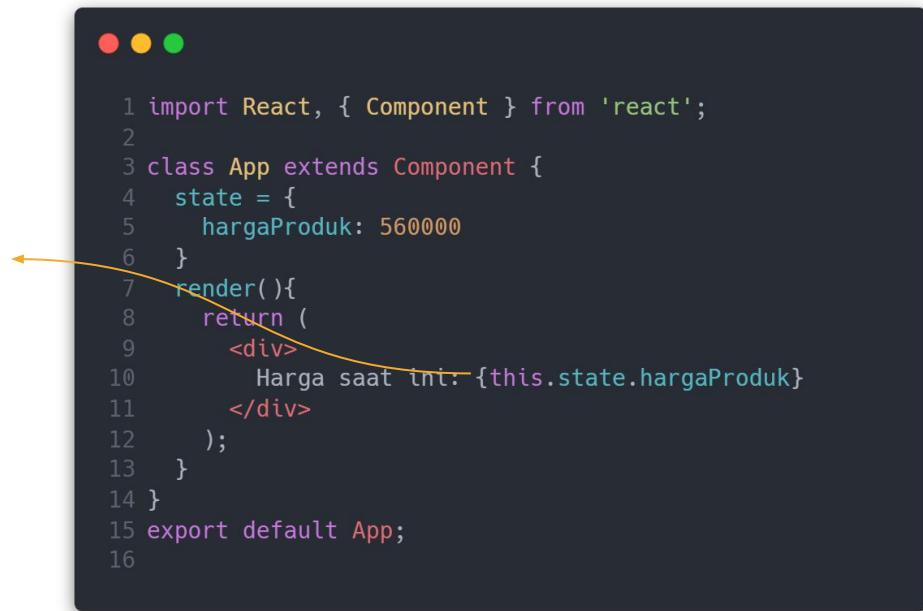


## Ini penerapan React.JS, guys!

Untuk bisa menampilkan '**Harga saat ini: 560000**', kamu perlu **memanggil variable 'this.state.hargaProduk'**.

Kenapa?

Karena variabel ini menampilkan value dari variabel state tersebut.



```
1 import React, { Component } from 'react';
2
3 class App extends Component {
4   state = {
5     hargaProduk: 560000
6   }
7   render(){
8     return (
9       <div>
10         Harga saat ini: {this.state.hargaProduk}
11       </div>
12     );
13   }
14 }
15 export default App;
16
```

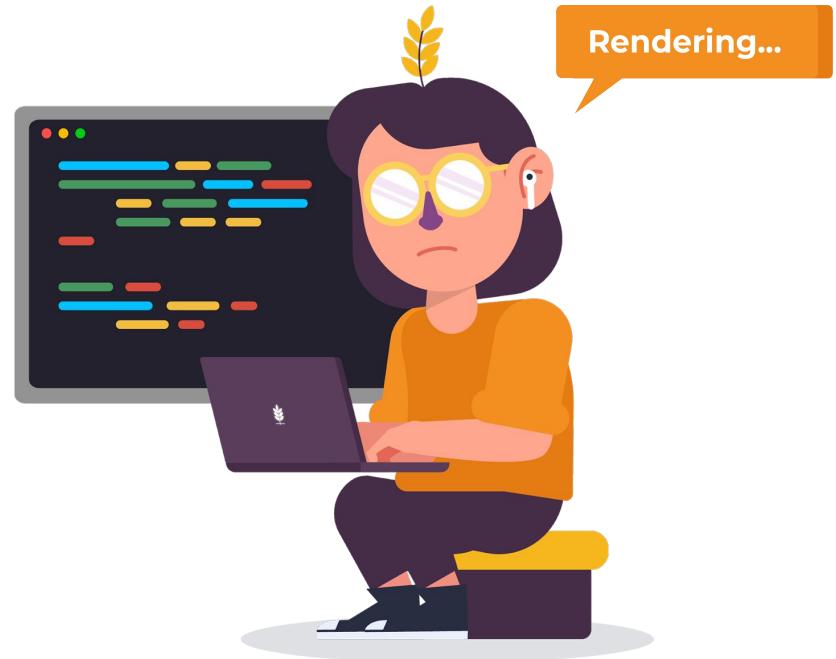


## Conditional Rendering

Conditional rendering adalah **suatu metode saat kamu mau menampilkan sebuah tampilan berdasarkan value pada nilai tertentu.**

Misalnya kamu mau menampilkan sebuah text 'Mahal' kalau variabel hargaProduk nilainya lebih dari 50000 dan ingin menampilkan text 'Murah' kalau nilainya kurang dari 50000.

Kalau begitu, kamu bisa pakai syntax seperti pada slide berikut:





### Coba perhatikan syntax ini ya, guys!

```
[ hargaProduk > 50000 ? <p>Mahal</p> : <p>Murah</p> ]
```

Melakukan perkondisian, jika kondisinya terpenuhi maka akan menghasilkan true. Jika nggak terpenuhi akan menghasilkan false.

Jika hargaProduk > 50000 menghasilkan true, maka `<p>Mahal</p>` akan dieksekusi

Apabila hargaProduk > 50000 menghasilkan false, maka `<p>Murah</p>` akan dieksekusi



## Nah, sekarang kita simulasikan aja yuk~

Coba kamu perhatikan deh! Pada situasi ini, syntax akan menampilkan '**Harga saat ini: Mahal**'. Kenapa bisa gitu ya? karena variable state hargaProduk **memiliki value 560000**. Value tersebut lebih besar dari 50000, sehingga perkondisian tersebut **menghasilkan nilai true**.

**Nah, apabila nilainya true**, maka kode setelah syntax tanda tanya (?) yaitu "Mahal", akan dieksekusi. Sebaliknya, kalau nilai yang dikondisikan false, maka kode setelah syntax titik dua (:) yaitu "Murah" yang akan dieksekusi.

```
 1 import React, { Component } from 'react';
 2
 3 class App extends Component {
 4   state = {
 5     hargaProduk: 560000
 6   }
 7
 8
 9   render(){
10     return (
11       <div>
12         Harga saat ini: {this.state.hargaProduk > 50000 ? "Mahal" : "Murah"}
13       </div>
14     );
15   }
16 }
17 export default App;
```



Coba cek pemahamanmu tentang ini bisa kali~ 😊

Nah, kalau **variable state hargaProduk memiliki value 50000**, maka akan menghasilkan nilai true atau false? lalu kode mana yang akan dieksekusi? Selamat berpikir! 😊

```
1 import React, { Component } from 'react';
2
3 class App extends Component {
4   state = {
5     hargaProduk: 560000
6   }
7
8
9   render(){
10     return (
11       <div>
12         Harga saat ini: {this.state.hargaProduk > 50000 ? "Mahal" : "Murah"}
13       </div>
14     );
15   }
16 }
17 export default App;
```



Habis belajar tentang rendering,  
saatnya kita belajar tentang  
**Component.**

Apa sih pentingnya component dalam  
penyusunan aplikasi React?





### Kita pahami komponen lewat analogi sepeda dulu yuk~

Sepeda nggak akan bisa kamu pakai buat long ride bareng komunitasmu kalau komponennya ada yang kurang. Yups, sepeda kan tersusun dari komponen seperti roda, rantai, fork, rem, dan lainnya, jadi kalau ada satu saja yang nggak terpasang, ya tentu sepeda nggak berfungsi dengan baik.

Sama saja kayak sepeda, **aplikasi React juga tersusun dari beberapa komponen penting untuk bisa beroperasi.**

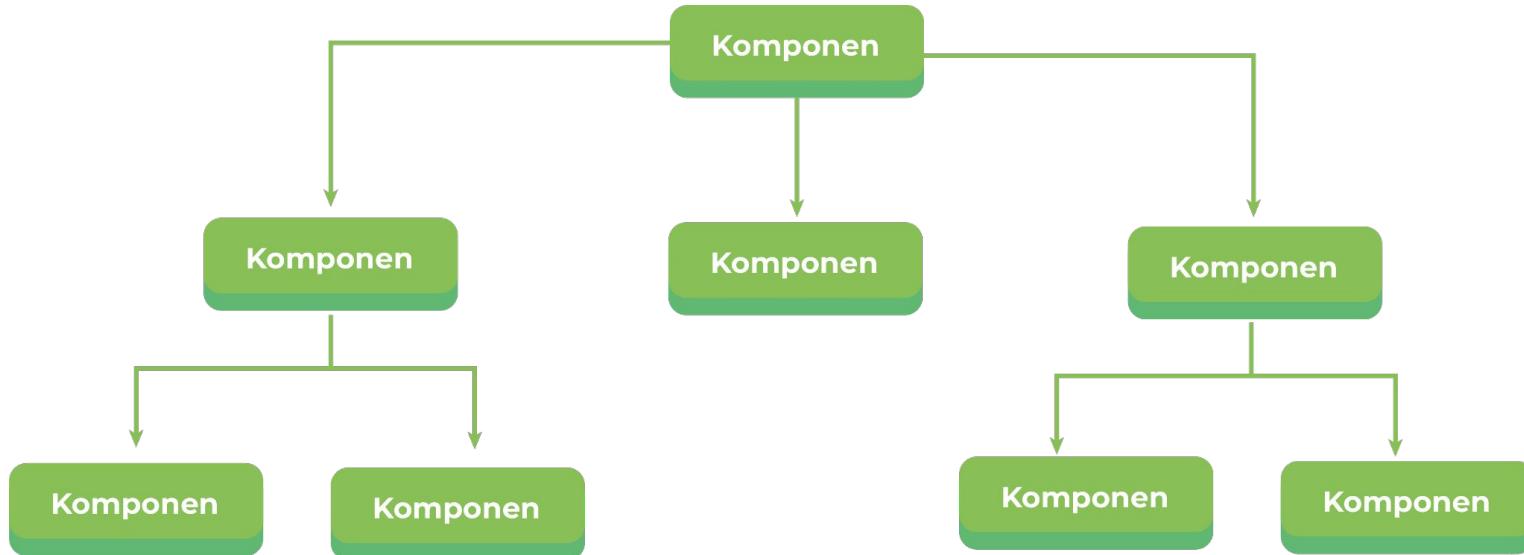




Kita juga bisa mengibaratkan **komponen React sebagai Lego**. Sebuah mainan Lego yang utuh terdiri dari beberapa komponen dan komponen-komponen tersebut bisa terdiri dari beberapa komponen juga, sampai ke komponen paling sederhana.



Berikut ini contoh pohon komponen (component tree), sebuah komponen besar bisa terdiri dari komponen yang lebih kecil dan seterusnya.

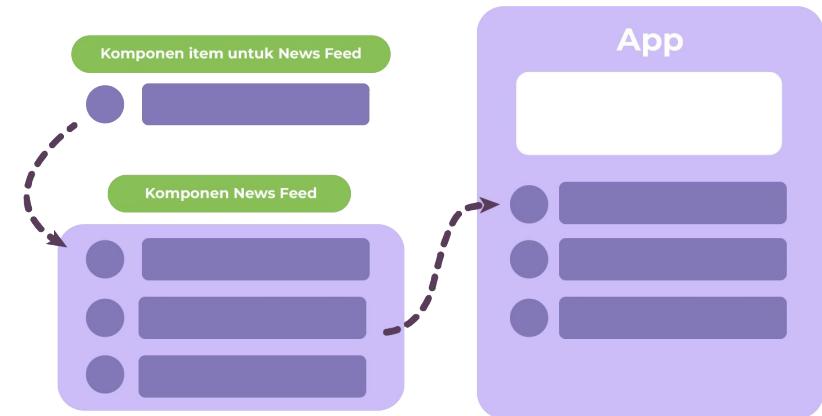


## React.JS punya beberapa macam komponen

Dari analogi sepeda dan Lego, kita tahu aplikasi React tersusun dari berbagai macam komponen, karena itulah komponen jadi bagian terpenting dari aplikasi.

React.JS punya beberapa macam component, yaitu:

- Stateful component
- Stateless component
- Function components
- Class components
- Dumb components
- Smart components.





### Kalau ada yang beda sama pacar kamu, pasti kamu berasksi kan?

Nah React.JS juga gitu~

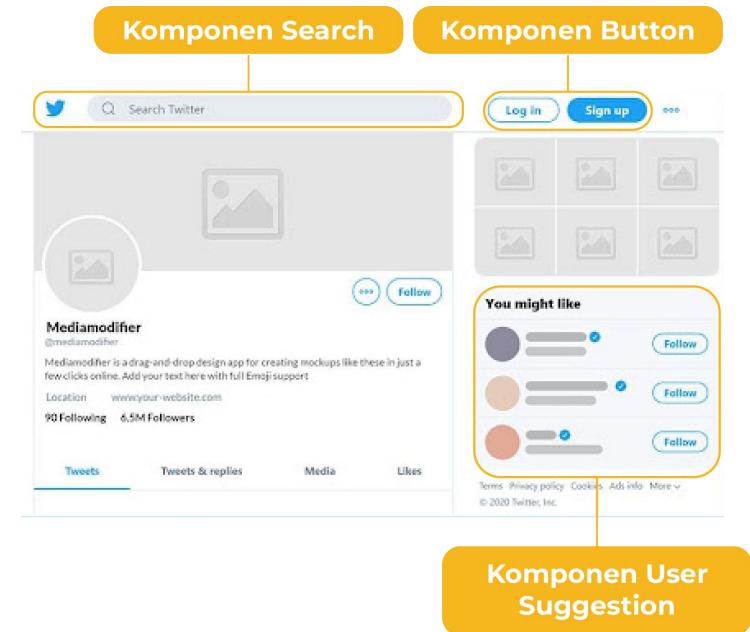
Pada dasarnya React.JS cuma merender komponen saat ada data yang berubah. Sesuai dengan namanya, “React”, maka ia **hanya akan berasksi saat ada perubahan data (reaktif)**.



## Lalu apa yang dimaksud dengan komponen pada React?

**Komponen** adalah **bagian-bagian** dari **UI**, contohnya seperti tombol, label, input, dll. Komponen juga bisa dibentuk dari komponen yang lain.

Sederhananya, dalam satu halaman website/aplikasi bisa terdiri dari banyak komponen.





### Apa tujuan dari pembuatan komponen pada React?

Tujuan dari pembuatan komponen React adalah **reusable code** (kode yang bisa digunakan kembali). Kode tersebut nggak hanya berisi JSX, tapi juga bisa berisi logika.

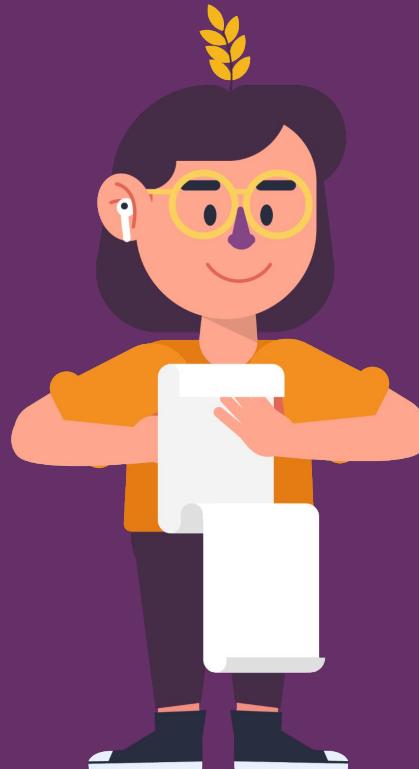
FYI, kode di dalam komponen React sifatnya independen dan terisolasi, sehingga data yang ada di dalamnya nggak akan dipengaruhi maupun mempengaruhi komponen lain, kecuali jika terjadi transaksi data antar komponen.





**Ngomong-ngomong, seperti yang  
sudah kita obrolkan tadi, React.JS  
punya beberapa jenis komponen!**

Di topik kali ini kita akan bahas dua  
jenis komponen React.JS berdasarkan  
cara membuat komponen di React.JS,  
Apa aja ya? Cekidot~





Ini dia dua jenis komponen yang bisa kamu pakai saat mau bikin komponen di React.JS, yaitu **Functional Component** dan **Class Component**.





## 1. Functional Component

Seperti namanya, functional component didefinisikan dengan kode **function**.

Component ini juga bisa disebut sebagai **Simple Component, karena tidak ada interaksi rumit** dan fitur component ini termasuk minimalis. Di dalam functional component, return harus berupa JSX. JSX ini pula yang nantinya akan dijadikan sebagai Elemen HTML.

nama komponen,  
harus dimulai dengan  
huruf kapital

```
function Hello(){  
  return <p>Hello</p>;  
}
```

Elemen JSX

<Hello />

Cara menggunakan  
komponen

[petanikode.com](http://petanikode.com)



## Begini contoh penulisan dari Function Component ya, guys!



```
import React from 'react'

export default function App() {
  return (<h1>Hello World</h1>)
}
```



## 2. Class Component

Kalau class component ya didefinisikan dengan class Javascript.

Class component **mendefinisikan tampilannya melalui method render**. Jadi return dari method render adalah JSX yang ingin ditampilkan.

```
nama komponen,  
harus dimulai dengan  
huruf kapital  
  
class Hello extends React.Component {  
  render(){  
    return <p>Hello</p>;  
  }  
}  
  
<Hello />  
Cara menggunakan  
komponen
```

class Component  
dari React.js

Elemen JSX

petanikode.com



### Begini contoh Penulisan dari Class Component ya, guys!

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

class HelloAgain extends Component {
  render() {
    return(<h1>Hello again world</h1>)
  }
}

export default HelloAgain;
```



Kamu juga akan butuh props dan state dalam pembuatan aplikasi agar antar component bisa berkomunikasi.

Karena itu, penting banget nih untuk paham dulu materi **Props dan State**.





## Props & State

Bagi React.JS, **props** adalah arguments atau input yang diteruskan oleh component lewat atribut HTML dan bersifat **read-only**.

Dengan adanya props, kamu bisa masukin nilai suatu variabel ke dalam HTML. Fitur ini nggak dimiliki oleh HTML biasa. Sekali lagi, ini mirip dengan view engine, jadi kita tinggal familiarkan sintaksnya yaa.

Sementara itu, **state** adalah data internal yang dapat diakses dan diubah oleh suatu class component dalam lifetime cycle tertentu.





## Props: functional/class component

Props dalam functional component perlu didefinisikan dahulu. Dalam code di samping definisi atribut props adalah type. Kemudian type akan menjadi variabel yang dapat ditampilkan di UI.

Untuk memasukkan value dari props, tambahkan atribut type beserta value ketika render component.



```
import React from 'react'  
export default function IniComponent(props) {  
  const { type } = props  
  
  return (  
    <h1>Ini component yang {type}</h1>  
  )  
}
```



```
<IniComponent type="small" />
```



## State: class component

State didefinisikan dalam class component. Nilai state bisa diakses dengan `this.state`, kemudian state bisa dimanipulasi dengan metode `this.setState()`.

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

class IniComponent extends Component {
  state = {
    count: 0
  }

  handleOnClick = () => {
    const { count } = this.state
    this.setState({ count: count + 1 })
  }
}
```

```
render() {
  return (
    <Fragment>
      <p>Total klik <strong>count</strong></p>
      <button onClick={this.handleOnClick}>Klik</button>
    </Fragment>
  )
}
export default IniComponent;
```



**React.JS adalah perpustakaan untuk bikin UI, karena itulah aspek styling juga jadi sorotan penting.**

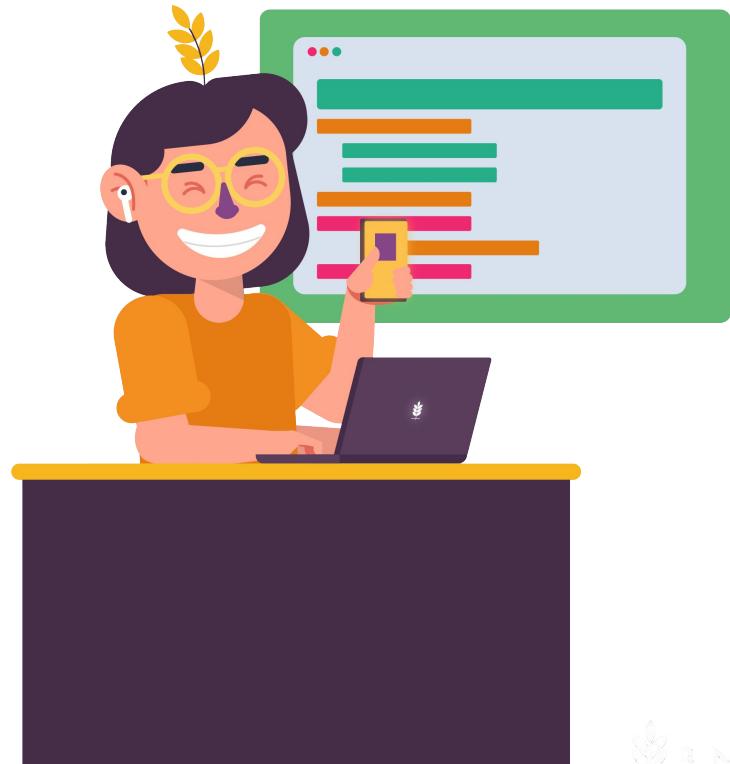




## Kayak rambutmu, program aplikasi juga butuh styling biar UI kece.

Oke, sebelumnya kan kamu sudah tahu beberapa jenis styling yaitu inline, internal, dan eksternal. Nah, dalam **React.JS ada dua jenis styling** lagi nih, yaitu:

1. Global
2. Module





## Styling: Global CSS

Global CSS adalah styling di mana setiap **deklarasi CSS di dalam file tersebut akan diterapkan di semua component** yang sesuai dengan selector-nya.

Contoh pemanggilannya seperti code di samping.



```
import React, { Component } from 'react'  
import ...  
  
// Ini Global Style  
import './App.css';  
  
// ... lanjutan code
```



## Styling: Module CSS

**CSS Module memungkinkan implementasi CSS secara parsial.** Kamu bisa pilih untuk override sebagian styling saja. Untuk melakukan hal ini, kamu perlu mengimpor CSS sebagai suatu variabel. Setelah itu, kamu implementasi styling dalam `className` suatu elemen HTML.

```
contoh.module.css

.btn {
  padding: 1em; border: 1px solid rgba(0,0,0,0.8);
  border-radius: 2px; background: none !important;
}
.btn-sm {
  font-size: 1em; padding: 1em;
}
.btn-success {
  background-color: green; color: white;
}
```

```
import React, { Fragment } from 'react'
import styles from './contoh.module.css'

export default function Contoh() {
  return (
    <Fragment>
      <button className={styles.btn}>Berubah</button>
      <button className="btn-success">Tetap</button>
      <button className="btn-sm">Tetap</button>
    </Fragment>
  )
}
```



**Component Lifecycle** akan jadi materi terakhir kita di topik ini biar kamu bisa menggunakan React.JS dengan baik.  
Apa itu component lifecycle?

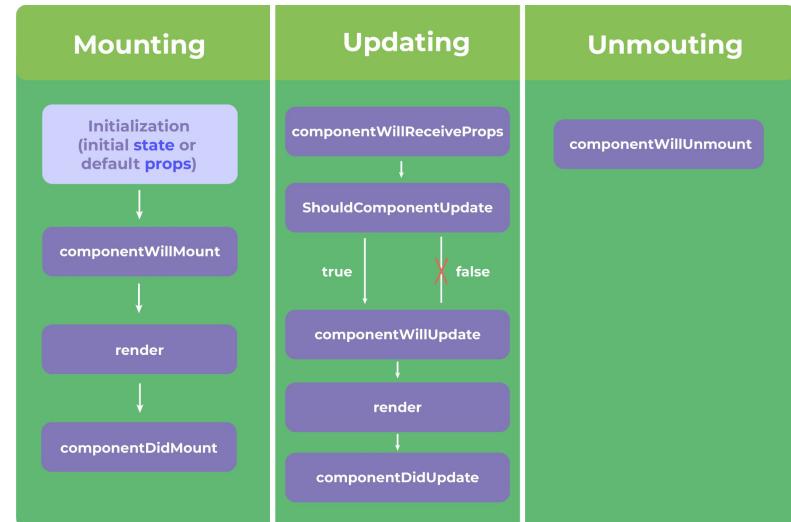




## Component Lifecycle, haloo? Apakah ini?

Secara sederhana, component lifecycle adalah **beberapa blok kode yang akan tereksekusi secara otomatis, tergantung waktu dan kondisinya.**

Kode-kode tersebut akan tereksekusi di dalam beberapa method bawaan React dan pada setiap fasenya memakai method-method tersendiri yang akan kita bahas lebih lanjut pada slide berikutnya.





Yash, ada beberapa fase siklus waktu yang terdapat pada setiap component di React.JS yaitu:

- Mounting Phase
- Updating Phase
- Unmounting Phase

Mari kita pelajari lebih lanjut tentang tiga fase ini.





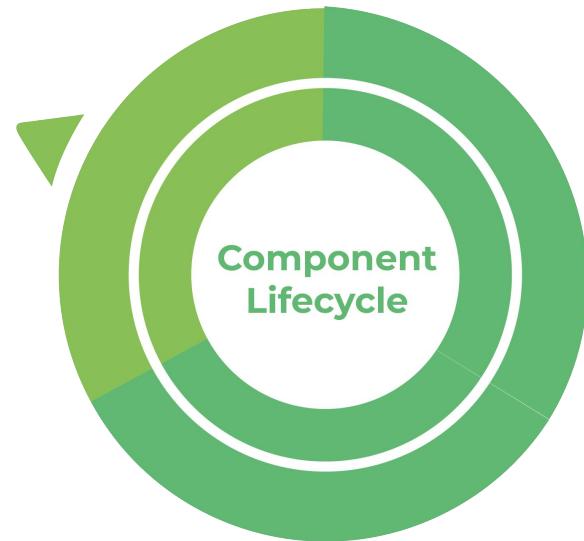
### Mounting Phase

Mounting phase adalah fase ketika halaman pada React.JS dibuka pertama kali.

Jadi ketika kamu membuka sebuah component atau halaman aplikasi di React, React akan mengeksekusikan tiga method bawaannya secara otomatis.

Ketiga method tersebut adalah **componentWillMount**, **render**, dan **componentDidMount**.

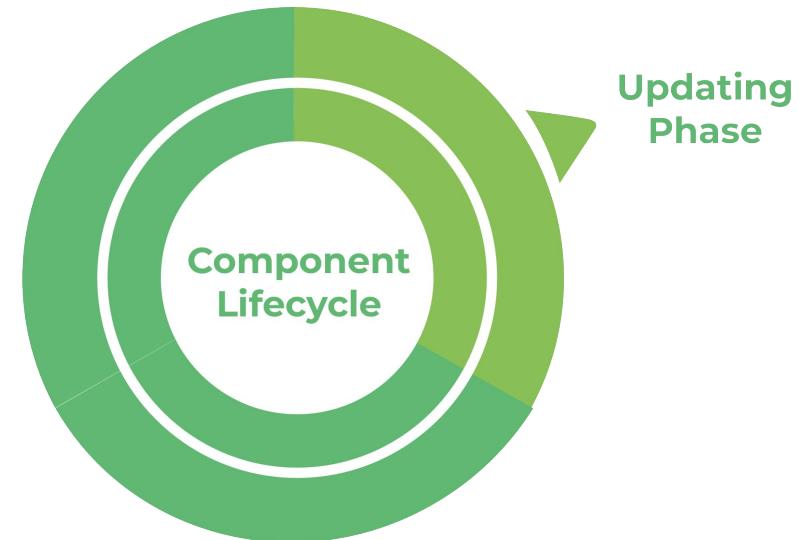
Mounting  
Phase





1. Ketika kamu buka sebuah component/page pada React, semua kode yang ada di dalam blok **componentWillMount** akan tereksekusi lebih dulu.
2. Setelah **componentWillMount** tereksekusi, maka React akan mengeksekusi semua kode yang berada pada method render.
3. Terakhir, React akan mengeksekusikan semua kode yang ada di dalam **componentDidMount**. Biasanya fase **componentDidMount** digunakan untuk bikin request data ke server.

```
1 import React, { Component } from "react"
2
3 class Header extends Component {
4   state = {
5     title: "this is title"
6   }
7
8   componentWillMount() {
9     console.log('Script di component will mount')
10  }
11
12   componentDidMount(){
13     console.log('Script di component did mount')
14  }
15
16   render(){
17     return(
18       <div>
19         <h1>{this.state.title}</h1>
20       </div>
21     )
22  }
23
24 export default Header
```



## Updating Phase

**Updating phase adalah fase saat tampilan UI pada component di-render ulang.** Hal ini terjadi karena adanya perubahan pada state atau props yang berkaitan dengan component tersebut.



1. Ketika kamu mengklik tombol, maka kamu meminta React untuk mengupdate value dari state 'title' menjadi 'Title baru'.

2. Setelah value state 'title' tadi berhasil diupdate, maka kode yang berada di dalam blok **componentDidUpdate** akan tereksekusi semuanya.

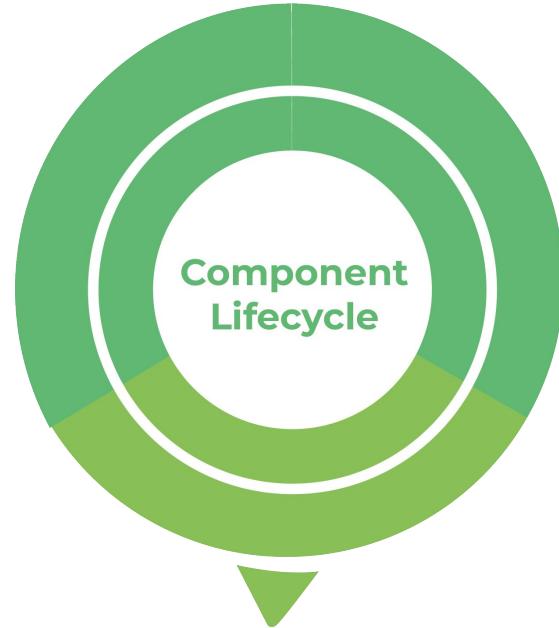
```
1 import React, { Component } from "react"
2
3 class Home extends Component {
4   state = {
5     title: "this is title"
6   }
7
8   componentDidUpdate(prevState, prevProps){
9     console.log('kode didalam componentDidUpdate')
10    }
11
12   render(){
13     return(
14       <div>
15         <h1>{this.state.title}</h1>
16         <button onClick={() =>{
17           this.setState({title: 'Title baru'})
18         }}>
19           Trigger Update State
20         </button>
21       </div>
22     )
23   }
24 }
25
26 export default Home
```



### Unmounting Phase

Unmounting phase adalah fase ketika kamu mau menutup component saat ini dan pindah ke component lain.

Misal saat ini kamu lagi buka component Home Page, lalu kamu mau pindah ke component Profile Page. Nah transisi perpindahan antara dua component tersebut akan memicu method componentWillUnmount pada component Home Page.



Unmounting  
Phase



Semua kode yang ada di dalam blok **componentWillUnmount** akan tereksekusi ketika kamu ingin berpindah dari Component Home ke Component lain.

```
1 import React, { Component } from "react"
2
3 class Home extends Component {
4   state = {
5     title: "this is title"
6   }
7
8   componentWillMount() {
9     console.log('Script di component will mount')
10  }
11
12
13  render(){
14    return(
15      <div>
16        <h1>{this.state.title}</h1>
17      </div>
18    )
19  }
20 }
21
22 export default Home
```

Saatnya kita  
**Latihan!**



Biar kamu makin paham dengan materi di sesi-sesi sebelumnya dan biar kamu makin semangat untuk mengeksplor lebih banyak hal terkait materi ini, kita latihan HTML dan CSS yuk, guys!

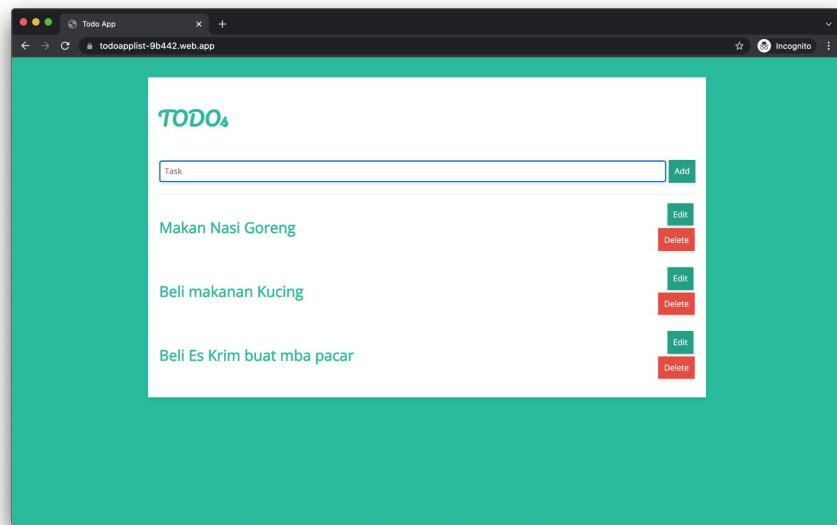
#### A. Instruksi

- Buatlah sebuah project dengan menggunakan create-react-app
- Beri nama project tersebut react-todo-app
- Setelah mengerjakan tugas, push folder react-todo-app ke remote-repository yang telah kamu buat.

## B. Tugas

Buatlah aplikasi Todo List menggunakan React.JS. Fitur-fitur aplikasi yang kamu buat ini akan sama persis seperti yang terdapat pada <https://todoapplist-9b442.web.app>

Lalu fitur tersebut dibagi menjadi 2 kategori, yaitu **Fitur Wajib** dan **Fitur Optional**



### Fitur Wajib

1. User bisa menambahkan To do Item pada Aplikasi dengan insert item pada sebuah form input.
2. Setiap item yang berhasil di insert akan langsung tampil di dalam sebuah list.
3. User bisa menghapus Item yang sudah ditambahkan.

### Fitur Optional

1. Data disimpan di **localStorage**.
2. User bisa mengedit item yang sudah ditambahkan.

Kamu bebas menggunakan desain dan mengubah tampilan aplikasi sesuai dengan selera kamu masing-masing. Untuk melihat referensi desain silahkan buka google ketikan cari '**todo app web design**'.

Semakin bagus tampilan dan desain yang kamu buat semakin bagus pula penambahan poin yang akan kamu dapat.

## Explore more!

Setiap hal yang kamu eksplor, pastikan kamu paham konsep nya dengan baik, jadi kamu nggak hanya sekedar copas aja dari referensi yang diberikan. Karena setiap hal yang kamu eksplor nantinya akan divalidasi pemahamannya. So, make sure kamu bener-bener paham konsepnya ya. Jangan asal copas ya, guys!

Berikut beberapa referensi bacaan dan video yang dapat kamu ikuti untuk menyelesaikan latihan ini. Good Luck!

1. <https://www.youtube.com/watch?v=9L2CSs2R2y4>
2. <https://www.youtube.com/watch?v=EIE08i2UJGI>
3. <https://javascript.plainenglish.io/build-a-simple-todo-app-with-react-561579b39ad1>
4. <https://betterprogramming.pub/how-to-build-a-todo-list-with-react-d2d5dd9f6630>

# Saatnya kita Quiz!





## 1. Pernyataan di bawah ini yang merupakan pernyataan yang salah adalah..

- A. Setiap halaman pada react hanya bisa memiliki satu komponen
- B. Setiap halaman pada react bisa menggunakan function komponen dan class komponen
- C. Setiap halaman pada react bisa memiliki lebih dari satu komponen



## 1. Pernyataan di bawah ini yang merupakan pernyataan yang salah adalah..

- A. Setiap halaman pada react hanya bisa memiliki satu komponen
- B. Setiap halaman pada react bisa menggunakan function komponen dan class komponen
- C. Setiap halaman pada react bisa memiliki lebih dari satu komponen

**Setiap halaman pada react dapat memiliki lebih dari satu komponen, tergantung kebutuhan pada halaman tersebut**



## 2. Apa kegunaan setState?

- A. Untuk menambahkan value pada state dan props
- B. Untuk mengupdate dan mengubah value state pada sebuah komponen
- C. Tidak ada gunanya



## 2. Apa kegunaan setState?

- A. Untuk menambahkan value pada state dan props
- B. Untuk mengupdate dan mengubah value state pada sebuah komponen
- C. Tidak ada gunanya

**setState** adalah sebuah metode yang terdapat pada class Component dan digunakan untuk mengubah value dalam sebuah komponen



### 3. Mana code yang tepat untuk menampilkan list variabel fruits

```
const fruits = [
  {
    id: 1,           name: 'Apple'
  },
  {
    id: 2,
    name: 'Banana'
  },
  {
    id: 3,
    name: 'Orange'
  }
]
```

A.



```
1 return (
2   <div>
3     {fruits.map(fruit => (
4       <p>{fruit.name}</p>
5     )))
6   </div>
7 )
```

B.



```
1 return (
2   <div>
3     {fruits.map(fruit => (
4       <p>{fruit.name}</p>
5     )))
6   </div>
7 )
```

C.



```
1 return (
2   <div>
3     {fruits.map(fruit => (
4       <p>{fruit.name}</p>
5     )))
6   </div>
7 )
```



### 3. Mana code yang tepat untuk menampilkan list variabel fruits

```
const fruits = [
  {
    id: 1,           name: 'Apple'
  },
  {
    id: 2,
    name: 'Banana'
  },
  {
    id: 3,
    name: 'Orange'
  }
]
```

A. ● ● ●

```
1 return (
2   <div>
3     {fruits.map(fruit => (
4       <p>{fruit.name}</p>
5     )))
6   </div>
7 )
```

B.

“name” adalah properti yang terdapat pada object setiap array.

C.



## 4. Apa kegunaan setProps?

- A. Untuk menambahkan value pada state
- B. Untuk mengupdate dan mengubah value props pada sebuah komponen
- C. Tidak ada gunanya



## 4. Apa kegunaan setProps?

- A. Untuk menambahkan value pada state
- B. Untuk mengupdate dan mengubah value props pada sebuah komponen
- C. Tidak ada gunanya

Tidak ada method yang Bernama props, ingat props tidak bisa di update, karena sifatnya read-only.



## 5. Tampilan apa yang akan muncul apabila “Tombol Hitung” di klik satu kali?

- A. Jumlah Angka :3
- B. Jumlah Angka :2
- C. Jumlah Angka: 6



```
1 import React, { Component } from 'react';
2
3 class App extends Component {
4   state = {
5     angka: 3
6   }
7
8   handleTombol() {
9     const angkaBaru = this.state.angka * 2;
10    this.setState({angka: angkaBaru});
11  }
12
13  render(){
14    return (
15      <div>
16        Jumlah angka: {this.state.angka}
17        <hr />
18        <button onClick={this.handleTombol}>Tombol Hitung</button>
19      </div>
20    );
21  }
22 }
23 export default App;
24
```



## 5. Tampilan apa yang akan muncul apabila “Tombol Hitung” di klik satu kali?

- A. Jumlah Angka : 3
- B. Jumlah Angka : 2
- C. Jumlah Angka: 6

Nilai awal state angka adalah 3, Ketika button “Tombol Hitung” diklik maka akan ada proses perhitungan nilai state Sekarang (3) dikali 2, hasilnya 6. Kemudia hasil perhitungan tadi akan diupdate ke state dan ditampilkan di UI



```
1 import React, { Component } from 'react';
2
3 class App extends Component {
4   state = {
5     angka: 3
6   }
7
8   handleTombol() {
9     const angkaBaru = this.state.angka * 2;
10    this.setState({angka: angkaBaru});
11  }
12
13  render(){
14    return (
15      <div>
16        Jumlah angka: {this.state.angka}
17        <hr />
18        <button onClick={this.handleTombol}>Tombol Hitung</button>
19      </div>
20    );
21  }
22}
23 export default App;
24
```

# Terima Kasih!



Next Topic

loading...