**Hari ke 9 - React : Hooks, State, Lifecycle Hooks, Handling event & Conditional rendering**

by Yogi Maha Putra

Pada Materi kali ini kita akan mempelajari :

* Apa itu React hooks ?
* State
* Lifecycle hooks ( useState )
* Handling Event
* Conditional Rendering

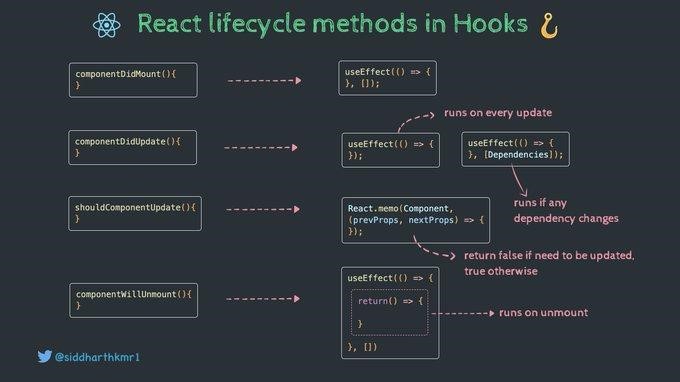
Berikut ini merupakan **Video Materi** dari Trainer untuk "React : Hooks, State, Lifecycle Hooks, Handling event & Conditional rendering"

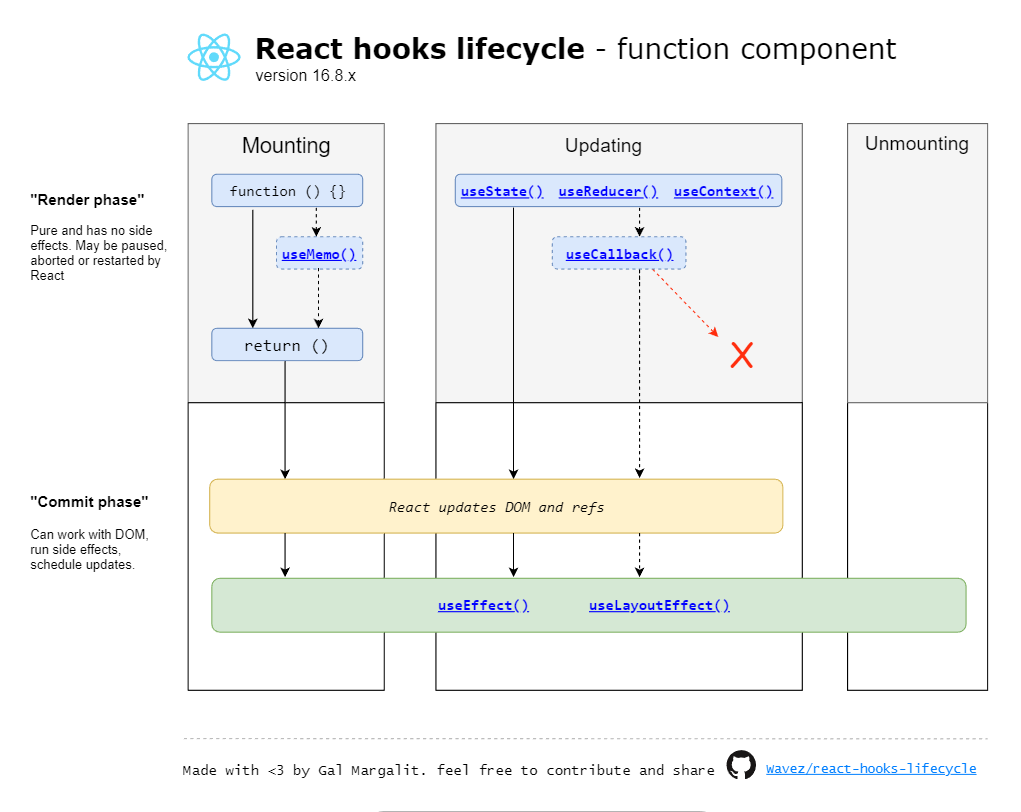
Apa Itu React Hooks ?

Hooks merupakan fitur baru di React 16.8 atau versi lebih baru dari itu. Dengan Hooks, kita dapat menggunakan state dan fitur React yang lain tanpa perlu menulis sebuah “class” baru. Hooks merupakan fungsi yang memungkinkan Anda untuk “mengaitkan” state dan fitur-fitur lifecycle React dari **function component**. Hooks tidak dapat berfungsi didalam “class” dan Hooks memungkinkan menggunakan React tanpa “class” serta memiliki syntax yang lebih **simple**tanpa harus memikirkan lagi penggunaan ( this ) maupun ( bind )

Components lifecycle di React Hooks

Berikut ini 2 Ilustrasi Lifecycle Method pada React Hooks :





Menurut Ilustrasi diatas, terdapat perbedaan yang cukup signifikan antara *lifecycle* method pada functional components dengan lifecycle di dalam class component, terutama pada bagian unmounting yang dimana pada React Hooks itu tidak ada method khusus. Namun, method functional component jika dibandingkan dengan class component, Kita hanya menggunakan useEffect saja untuk beberapa Lifecycle Component. Berikut ini merupakan beberapa method yang tersedia pada Functional Component ( React Hooks ) :

* useState
* useEffect
* useRef
* useLayoutEffect
* useCallback
* useMemo
* useReducer
* useContext

Pada Materi React Hooks ini, kita tidak akan mempelajari semuanya melainkan hanya beberapa saja seperti :

* useState
* useEffect
* useContext

**Sebelum kita masuk pada penjelasan component lifecycle, Kita harus mengetahui dulu tentang state.**

😬 Apa itu State ?

State merupakan data yang tersimpan dalam sebuah component. State bersifat private dan hanya relevan terhadap component itu sendiri. Berbeda dengan props yang valuenya dilempar dari component lain, state justru dapat menyimpan dan mengubah datanya sendiri dari dalam component. Contohnya  :

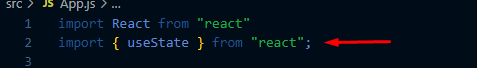
Pada React Hooks ( functional Component )

Const [stateCount, setStateCount] = useState(0)

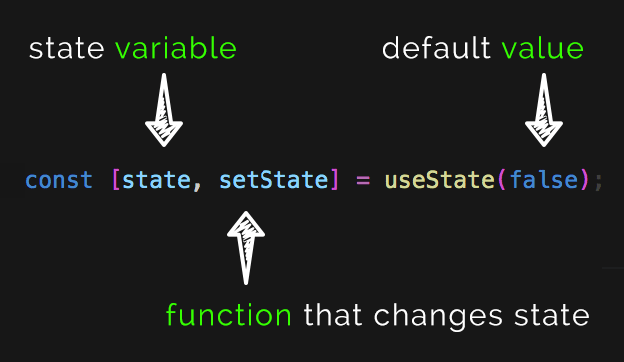
**Mari Kita ulik Method Lifecycle React Hooks**

**useState()**

Jangan lupa, setiap menggunakan state kita harus import terlebih dahulu :



Berikut ini pola state pada useState() :



Seperti penjelasan diatas tadi, React hooks useState digunakan untuk membuat state di function komponen. Berikut ini cara menggunakan useState Pada functional Component :

const [pesertaLomba, setPesertaLomba] =  useState([ 'Budi', 'Susi', 'Lala', 'Agung' ])  
const [inputName, setinputName]  =  useState("")  
const [indexOfForm, setIndexOfForm] =  useState(-1)

Pada contoh diatas, kita menggunakan useState. useState adalah sebuah Hooks, dan akan mengembalikan nilai dari state (pada contoh diatas: pesertaLomba) dan fungsi yang bisa kita gunakan untuk mengubah nilai tersebut (setPesertaLomba). Fungsi setPesertaLomba disini mirip dengan cara kerja this.setState pada method class component***.***

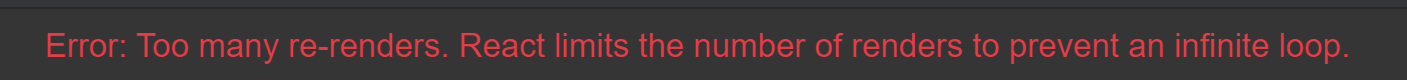
Jika kita bandingkan dengan insialisasi state pada class component :

constructor(props) {  
    super(props);  
    this.state = {  
      name: "John",  
      age: 21,  
    };  
  }

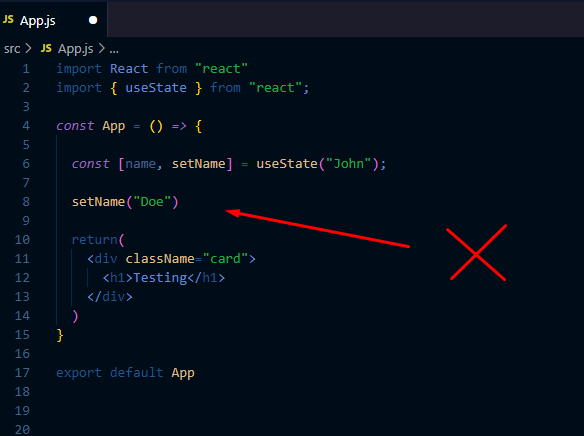
Berbeda dengan this.state*,*state di dalam useState() tidak harus berbentuk *object*. Dan kita bisa menggunakan lebih dari satu useState di dalam satu component, contohnya lagi seperti ini:

const [name, setName] = useState("John");  
const [age, setAge] = useState(21);  
const [job, setJob] = useState({  
 jobTitle: "UI/UX Designer",  
 company: "UNIKOM CodeLabs"  
});

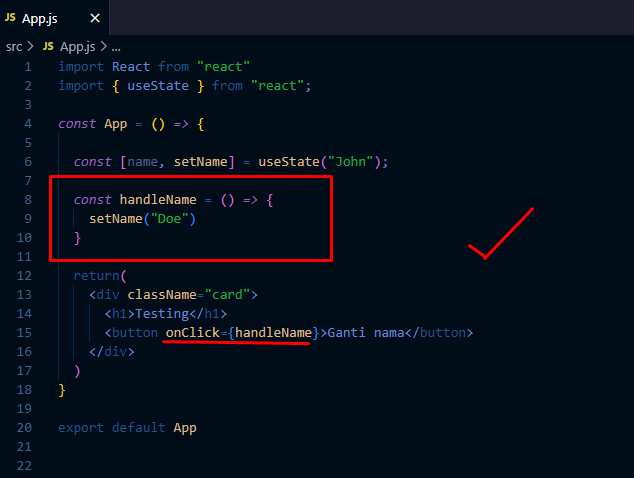
**PERINGATAN :**jangan memanggil setState() diluar function/tanpa dibungkus function untuk menghandle karena akan mendapatkan error seperti dibawah ini :



Contoh **Salah** :

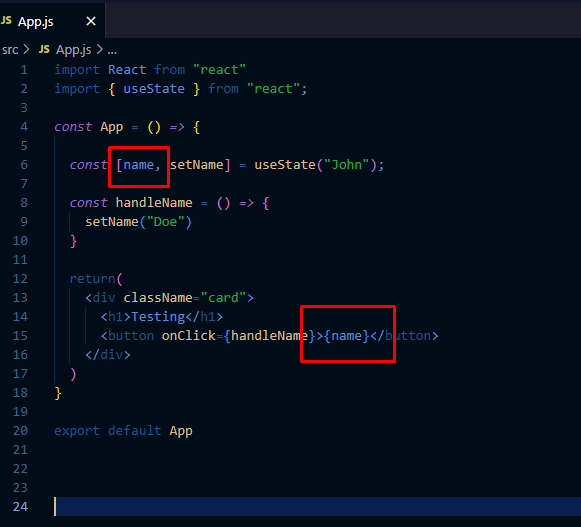


Contoh **Benar,**Harus melewati prosess **handling** terlebih dahulu atau bisa juga **didalam useEffect** :



Apa itu attriubte **onClick**?? Apa itu **Handler**?? Kita akan pelajarin pada materi di bawah nanti.

Bagaimana memanggil state didalam jsx ??



Setiap memanggil kode javascript kedalam jsx, selalu sematkan '{ kode... }' bracket seperti ini.

Berikut ini soure code contoh penerapan useState diatas pada react Hooks :

|  |
| --- |
| import React from "react" import { useState } from "react";  const App = () => {  const [name, setName] = useState("John");   const handleName = () => {    setName("Doe")   }   return(    <div className="card">       <h1>Testing</h1>       <button onClick={handleName}>{name}</button>     </div>  ) }  export default App |

**useEffect() : ( Materi Pekan 3 Hari 1 )**

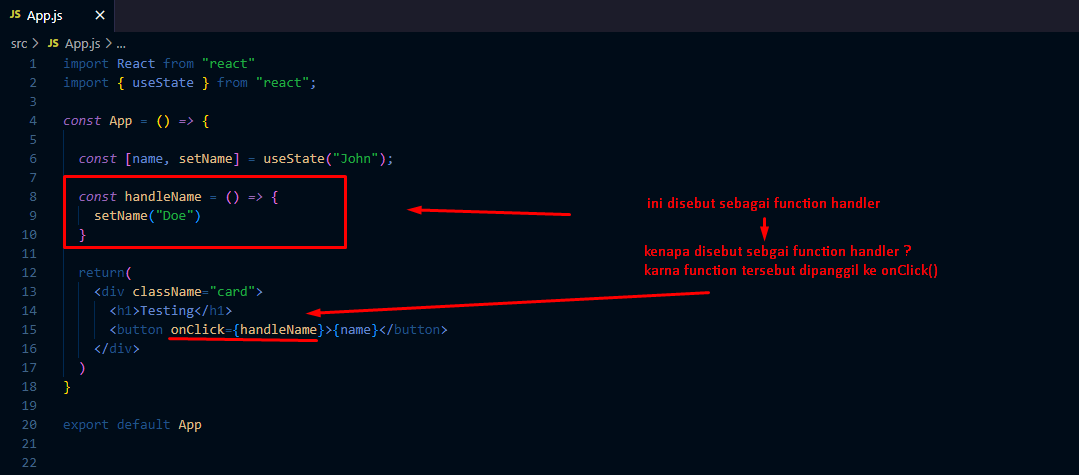
**useContext() : ( Materi Pekan 3 Hari 5 )**

💒 Apa itu Handler ?

Menangani *events handling* dengan elemen React sangat mirip seperti menangani sebuah *events* pada elemen DOM. Ada beberapa aturan :

* *Events handling* pada React biasanya ditulis dalam bentuk *camelCase*, bukan *lowercase*.
* Dengan JSX Anda dapat mengoper *function* sebagai *event handler*, bukan sebagai *string*.

Berikut ini polanya seperti contoh sebelumnya :



Tidak akan selalu ke attribute **onClick**untuk mengimplementasi handler, ada juga **onSubmit**, **onSearch**dan yang lainnya. nanti lambat laun kita akan mengetahui penggunaan handler dimana saja yang tepat 👍

🍋 Belajar Conditional Rendering

Masih ingat pada konsep *conditonal javascript* ?

Ya benar, yang polanya seperti ini :

|  |
| --- |
| if(kondisi/indikator){    .....      }     else{     .....     } |

Nah konsep ini akan kita **padukan/gabungkan**dengan **jsx**dalam sebuah component.

Begini kurang lebih :

|  |
| --- |
| if(kondisi/indikator){    ..... html jsx 1     }     else{     .... html jsx 2    } |

Jadi pada konsep ini dia akan merender jenis jsx mana (html jsx 1 atau html jsx 2 ) kedalam return component. akan tetapi, conditional yang kita terapkan pada return bukan lah *if else ,*melainkan *tenary operator :*

|  |
| --- |
| kondisi/indikator ? "kondisi true // mirip seperti if" : "kondisi false //  mirip seperti else" |

seperti ini contohnya :



Penjelasan pola kode :



**Referensi**:

* <https://id.reactjs.org/docs/hooks-intro.html>
* <https://id.reactjs.org/docs/hooks-overview.html>
* <https://id.reactjs.org/docs/hooks-state.html>
* <https://id.reactjs.org/docs/hooks-effect.html>
* <https://id.reactjs.org/docs/handling-events.html>
* <https://reactjs.org/docs/conditional-rendering.html#gatsby-focus-wrapper>