

# COACHING PROGRAM Day 2

## React.js

### Bagian 0: Pengenalan React.js

[React.js](#) adalah sebuah library JavaScript yang digunakan untuk membangun antarmuka pengguna (UI) pada aplikasi web. React dikembangkan oleh Facebook dan kemudian diopen-source-kan. React digunakan secara luas untuk mengembangkan aplikasi web yang dinamis dan responsif serta performa yang bagus karena mendukung konsep reusable component dan penerapan [SOLID](#) principle.

Selain itu, ada beberapa framework yang menggunakan react sebagai dasar, namun memiliki fitur-fitur unik lainnya, seperti [Angular](#) yang memudahkan dalam penulisan kode yang bersifat asynchronous, [Vite.js](#) yang mendukung kecepatan dalam kompilasi dan pengembangan, [Next.js](#) yang mendukung konsep Server dan Client Side Rendering untuk meningkatkan [SEO](#), dan framework-framework lain yang memiliki fitur dan kegunaan masing-masing.

### Bagian 1: React Component

#### Pengenalan React Component

Dalam React, komponen adalah blok dasar pembangun yang digunakan untuk membangun antarmuka pengguna (UI) dalam aplikasi web. Komponen adalah unit dasar dalam pengembangan dengan React dan memungkinkan Anda untuk mengelola UI secara terstruktur.

```
const UserCard = (props) => {  
  
  const umur = 2023 - props.tahunLahir;  
  
  return (  
    <div>  
      <div>Nama: {props.nama}</div>  
      <div>Umur: {umur}</div>  
    </div>  
  )  
}  
  
export default UserCard
```

```
const UserCard = (props) => {  
  
  const umur = 2023 - props.tahunLahir;  
  
  return (  
    <div>  
      <div>Nama: {props.nama}</div>  
      <div>Umur: {umur}</div>  
    </div>  
  )  
}  
  
export default UserCard
```

Diagram annotations for the code above:

- Nama Component**: Points to `UserCard` in the function definition.
- Property**: Points to `props` in the function definition.
- Deklarasi variabel**: Points to `const` in the function definition.
- Memanggil props tahunLahir**: Points to `props.tahunLahir` in the calculation of `umur`.
- Elmen yang di kembalikan**: Points to the `return` statement.
- Memanggil props nama**: Points to `props.nama` in the JSX element.
- Memanggil variable umur**: Points to `umur` in the JSX element.
- Mengekspor komponen agar dapat dipakai di file berbeda**: Points to `export default`.

- **Nama Component**: Nama dari komponen yang ingin di return, bersifat bebas namun **wajib** berawalan huruf kapital, contoh: UserCard, SubmitButton, LoginForm, dll.
- **Property**: Suatu mekanisme untuk mengirimkan data dari **pemanggil komponen** pada **komponen yang dipanggil**. Dari contoh diatas, element dipanggil dengan code seperti ini: `<UserCard nama="John" tahunLahir="2003" />`

## COACHING PROGRAM Day 2

### React.js

- **Deklarasi variabel:** Pada react, variabel di deklarasi di dalam komponen, namun sebelum "return", karena return adalah batas code yang akan di eksekusi dan kemudian dikembalikan kepada pemanggil.
- **Memanggil Props:** props atau property bertipe Object sehingga diakses menggunakan titik dan diikuti dengan nama propertynya.
- **Element yang dikembalikan:** Walaupun React adalah framework Javascript, tidak sepenuhnya code akan ditulis dalam bahasa javascript. Nilai return ini biasanya berisikan baris kode yang berformat HTML.
- **Memanggil Props dalam elemen yang akan dikembalikan:** Pada contoh diatas, property nama dan variabel umur dipanggil pada format HTML. Bedanya dengan HTML biasa, React dapat langsung memanggil variabel tersebut dengan menggunakan curly bracket { }.
- **Mengeksport Komponen:** React mendukung modularitas yang tinggi, konsep SOLID, dan sangat mendukung untuk mengurangi duplikasi kode dan penggunaan ulang komponen. Oleh karena itu, komponen di taruh di satu file khusus yang dapat dipakai oleh berbagai komponen lainnya.

### React Hooks

React Hooks adalah fitur yang diperkenalkan dalam React 16.8 untuk memungkinkan penggunaan state (keadaan) dan fitur-fitur lainnya dalam komponen fungsi, yang sebelumnya hanya dapat dilakukan dalam komponen kelas. Hooks memungkinkan pengembang untuk menangani keadaan dan efek samping dalam komponen fungsi, yang membuat kode lebih bersih dan lebih mudah dimengerti. Ada beberapa Hooks bawaan yang disertakan dengan React, seperti useState, useEffect, useContext, dan lainnya.

**useState:** adalah Hook yang digunakan untuk menampung variabel yang dapat berubah karena suatu kondisi, yang disebut sebagai *variabel reactive*. Hook inilah yang paling sering digunakan karena memudahkan dalam pengembangan website yang interaktif. Contoh:

```
import { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  )
}
```

Count: 0

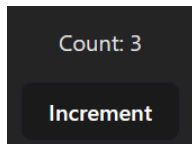
Increment

Inisialisasi untuk hook useState mirip inisialisasi variabel, namun memiliki 2 bagian, pada contoh: [count, setCount], artinya **count** adalah nama variabel, dan **setCount** adalah fungsi untuk mengubah variabel **count**. Setelah itu diikuti dengan deklarasi nilai: useState(0), yang artinya menggunakan hook useState, dengan nilai default = 0 untuk variabel count.

## COACHING PROGRAM Day 2

### React.js

Setelah itu, variabel **count** dipanggil pada komponen: `{count}`, dan terdapat komponen **button** sebagai pengubah variabel tersebut. Komponen button menggunakan property **onClick**, yang artinya fungsi akan dipanggil saat button itu di klik. Contoh saat di klik 3x:



[useEffect](#): adalah Hook yang menjalankan kode setelah proses render komponen selesai.

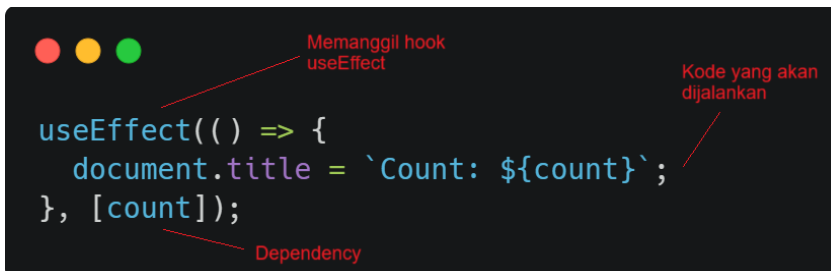
Contoh:

```
import { useState, useEffect } from 'react';

function Example() {
  const [count, setCount] = useState(0);

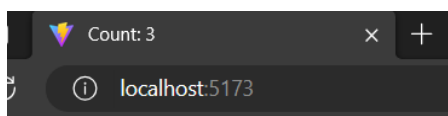
  useEffect(() => {
    document.title = `Count: ${count}`;
  }, [count]);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

A code snippet showing the `useEffect` hook. Red arrows point to different parts of the code with labels: 'Memanggil hook useEffect' points to `useEffect`, 'Kode yang akan dijalankan' points to the function body `{ document.title = `Count: ${count}`; }`, and 'Dependency' points to the dependency array `[count]`.

```
useEffect(() => {
  document.title = `Count: ${count}`;
}, [count]);
```

Pada contoh diatas, kode `document.title = `Count: ${count}`` dipanggil saat terjadi perubahan pada variabel **count**, karena variabel **count**, ditaruh pada tempat dependency. Kode tersebut berfungsi untuk merubah title pada web sesuai nilai variabel count.



Untuk dependency, tidak wajib diisi, dan menyesuaikan kebutuhan. Jika dependency tidak diisi, maka kosongkan saja arraynya seperti ini `[]` dan kode akan hanya dijalankan 1x saat komponen di load.

Selain kedua hook diatas, ada banyak react hooks lain yang memiliki kegunaan dan fungsi yang berbeda, namun hanya menggunakan dua hook diatas, kalian sudah dapat membuat sebuah website fullstack yang interaktif.

# COACHING PROGRAM Day 2

## React.js

### Form Handling

Form Handling adalah salah satu aspek penting dalam pengembangan aplikasi web, termasuk aplikasi React. Form handling mengacu pada cara Anda mengelola input dari pengguna melalui formulir dalam aplikasi Anda, serta cara Anda memproses dan mengirim data yang dikumpulkan dari formulir tersebut.

1. **Elemen Form:** dalam React mirip dengan elemen HTML, seperti `<form>`, `<input>`, `<textarea>`, dan `<select>`. Anda menggunakan elemen-elemen ini untuk membuat formulir dalam aplikasi React.
2. **State Management:** Seperti yang sudah dipaparkan sebelumnya, dalam form handling, dibutuhkan tempat untuk menampung variabel dari inputan user, sehingga diperlukannya peran hook **useState** dalam menangani hal ini.
3. **Penanganan Perubahan:** Anda perlu menangani perubahan nilai dalam input formulir. Ini biasanya dilakukan dengan menggunakan event handler seperti `onChange`. Ketika nilai berubah, Anda akan memperbarui state dengan nilai baru. Contoh:

```
function App() {  
  const [name, setName] = useState('');  
  
  const handleSetName = (event) => {  
    setName(event.target.value);  
  };  
  
  return (  
    <div>  
      <div>Hello {name}!</div>  
      <input type="text" value={name} onChange={handleSetName} />  
    </div>  
  )  
}
```

**Inisialisasi:** Dari contoh diatas, terdapat inisialisasi variabel **name** yang menggunakan hook `useState` dengan nilai default `"` yang artinya bertipe string namun kosong.

**Handler:** ada fungsi **handleSetName** yang bertugas sebagai handler (penangan) untuk fungsi **setName**. Handler ini tidak bersifat wajib, karena `setName` dapat langsung dipanggil pada tag input, namun lebih direkomendasikan untuk menggunakannya agar komponen HTML tetap terlihat bersih sehingga memudahkan proses development.

**Parameter:** terdapat parameter pada handler tersebut dengan nama **event**. Parameter ini bertugas menangkap setiap event (kejadian) yang terjadi saat `onChange` (terjadi perubahan). Event ini memiliki property yang banyak, dan untuk mengakses String yang diinput user, selalu menggunakan **event.target.value**. Parameter ini bisa menggunakan nama yang bebas, namun biasanya dinamakan **event** atau **e** saja untuk keseragaman dalam penamaan.

4. **Property onChange:** untuk property `onChange`, jika fungsi yang dipanggil itu berasal dari setter `useState` (contoh diatas yaitu **setName**), maka formatnya akan seperti ini `onChange={(event) => setName(event.target.value)}`. Dibutuhkan `() => ...` sebelum

## COACHING PROGRAM Day 2

### React.js

memanggil fungsi karena fungsi tersebut berasal dari setter useState. Namun jika memanggil fungsi yang dideklarasikan manual (seperti contoh diatas: `const handleSetName = () => { ... }`), maka pemanggilan pada **onChange**, tidak membutuhkan `() => ...` pada awalan, sehingga dapat langsung dipanggil `onChange={handleSetName}`.

### Naming Convention

Setelah melihat contoh kode pada Form Handling, ada beberapa poin yang dapat dipetik terkait penamaan suatu variabel dan fungsi, yaitu:

1. Gunakan kata yang mendefinisikan isi dari variabel tersebut. Contoh di form handling yaitu ada variabel **name** yang isinya adalah nama yang berasal dari inputan user.
2. Pada naming convention untuk useState, setter diawali dengan kata **set-** dan diikuti dengan nama variabel dengan format [Camel Case](#) (ini adalah contoh Camel Case), sehingga jadi **setName**. Camel Case diawali dengan huruf kecil, dan huruf kapital setiap kata berikutnya dan sangat umum digunakan dalam penamaan variabel dan fungsi.
3. Fungsi handler, pada contoh Form handling, terdapat handler yang bertugas untuk memanggil fungsi setName dari deklarasi useState. Untuk handler seperti ini, biasanya diawali dengan kata **handle-** dan diikuti dengan tujuan dari fungsi ini. Jadi bisa saja **handleSetName**, atau **handleNameChange**, karena dua-duanya memiliki arti yang sama. Dan untuk formatnya juga sama seperti sebelumnya, menggunakan camel case.