COACHING PROGRAM Day 3
React.js

Bagian 2: Consume API

Pengertian Consume API

Mengonsumsi (atau "consume") API merujuk pada proses penggunaan atau akses terhadap antarmuka pemrograman aplikasi (API) yang disediakan oleh layanan atau sistem eksternal untuk mengambil atau mengirim data. API adalah kumpulan aturan dan protokol yang memungkinkan berbagai perangkat lunak atau aplikasi untuk berkomunikasi satu sama lain. Saat Anda mengonsumsi API, Anda menggunakan permintaan HTTP (seperti GET, POST, PUT, atau DELETE) untuk berinteraksi dengan API eksternal untuk mengakses sumber daya atau layanan tertentu.

Axios

Pada dasarnya, API dapat di consume oleh fungsi bawaan dari javascript, yaitu fetch(), namun fungsi ini memiliki cukup banyak kekurangan dalam melakukan consume API. Axios adalah salah satu framework yang paling banyak digunakan dalam API consume karena kemudahan dalam sintaks, fitur yang beragam, penanganan error, intersepsi permintaan-respon, mendukung CORS, dan lain-lain.

Untuk instalasi Axios cukup mudah, jalankan perintah `npm i axios` pada folder project react yang telah dibuat pada materi pre-day 2 menggunakan vite.js.

```
C:\Users\Fanes\test\UKM Programming>npm i axios

added 9 packages, and audited 281 packages in 2s

98 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Sintaks axios untuk melakukan request cukup simpel, misal kita akan melakukan fetching/consume kepada API: https://pokeapi.co/api/v2/pokemon/1.

```
useEffect(() => {
  const fetchData = async () => {
    const res = await axios.get("https://pokeapi.co/api/v2/pokemon/1");
    console.log(res);
  }
  fetchData();
}, []);
```

Code

Pada code diatas, gunakan useEffect untuk menjalankan fungsi yang melakukan fetching kepada API diatas. Dalam hook useEffect tersebut, terdapat satu fungsi bernama **fetchData** yang dideklarasi menggunakan arrow function yang asynchronous.

COACHING PROGRAM Day 3
React.js

Apa itu asynchronous?, simpelnya async/await adalah fitur dari javascript yang membuat suatu baris code itu menunggu suatu proses hingga eksekusi selesai (await), baru melanjutkan baris code berikutnya.

Async dapat di deklarasi seperti diatas bila menggunakan arrow function, atau bila menggunakan function biasa: **async function fetchData() { // … }**.

Pada contoh diatas, dalam fungsi fetchData, terdapat deklarasi variabel bernama **res** dengan nilai **axios.get("…")** yang di **await**. Sehingga proses axios dalam melakukan fetching itu ditunggu hingga selesai, baru melanjutkan ke baris selanjutnya, yaitu **console.log(res)** dari hasil tersebut. Hal ini dibutuhkan dalam fetching karena proses fetching ini bersifat dinamis yang bergantung oleh kecepatan internet masing-masing user yang mengakses, sehingga sistem tidak dapat memprediksi kapan proses tersebut selesai.

Kenapa async tidak dideklarasi di useEffect seperti ini?

CONTOH YANG SALAH

```
useEffect(async () => {
  const res = await axios.get("https://pokeapi.co/api/v2/pokemon/1");
  console.log(res);
}, []);
```

Singkatnya, react melarang hal tersebut karena hook useEffect tidak dapat melakukan pengembalian terhadap suatu Promise Task (fetching data). Oleh karena itu fetching dilakukan di fungsi yang dideklarasi, dan useEffect hanya bertugas memanggilnya saja. Contoh bentuk lain dalam melakukan fetching:

```
const fetchData = async () => {
  const res = await axios.get("https://pokeapi.co/api/v2/pokemon/1");
  console.log(res);
}
useEffect(() => {
  fetchData();
}, []);
```

Pada contoh diatas, fungsi fetchData dideklarasi diluar useEffect, dan useEffect hanya melakukan pemanggilan fungsi saja (fetchData();), kedua cara ini tidak ada perbedaan signifikan dan terserah kalian mau menggunakan yang mana.

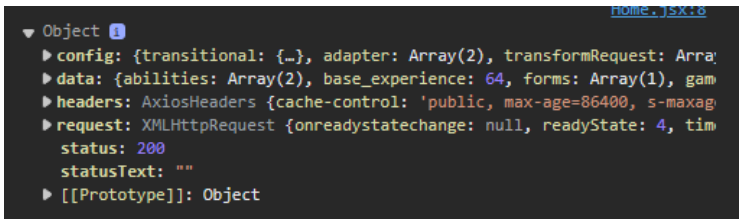
Note: jangan lupa melakukan import module axios menggunakan kode :
`import axios from "axios";`
dan ditaruh pada bagian atas file.

COACHING PROGRAM Day 3
React.js

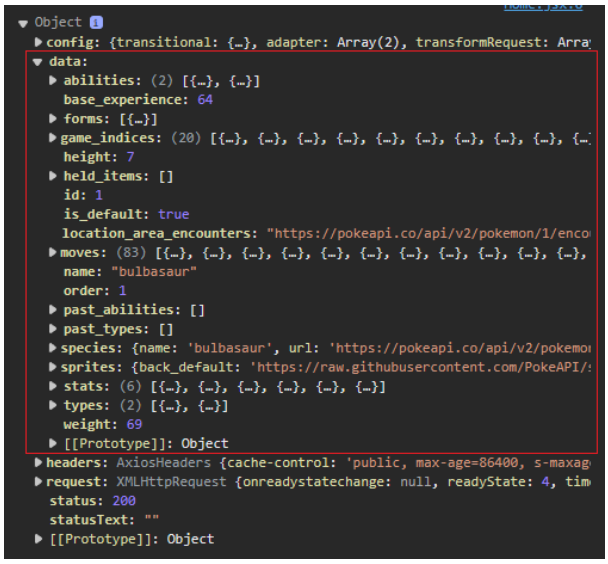
Membaca Hasil dari fetching

Bila dilihat pada console browser (silahkan cari cara membuka console pada browser masing-masing), dapat dilihat bahwa terdapat dua baris yang di print.

Hal tersebut normal pada tahap development react karena strict mode. Kalian dapat mengabaikan hal tersebut dan jangan heran, karena pada tahap development, semua code akan dijalankan 2x untuk hal tertentu.



Bila kalian buka salah satu hasil print tersebut, dapat dilihat bahwa hasil return bertipe Object yang memiliki property config, data, headers, requests, status, dan statusText. Normalnya, data hasil fetch API disimpan pada property data:



Dalam property data, terdapat banyak property anakan lagi yang biasa disebut nested object. Dapat dilihat bahwa pada property data ini, terdapat property **name** dengan value “bulbasaur”, yang artinya bahwa API ini memiliki data-data untuk pokemon dengan nama “bulbasaur”.

Untuk mengakses nama dari pokemon tersebut, kalian dapat menggunakan kode **res.data.name**. Kode inilah yang nantinya akan ditampilkan pada halaman web menggunakan hook useState.

COACHING PROGRAM Day 3
React.js

Ekstrak Data API ke State

Pada pertemuan day 2, terdapat folder project vite.js dengan nama src dan paste code diatas ke file App.jsx.

```jsx
import axios from 'axios'
function App() {

  const fetchData = async () => {
    const res = await axios.get("https://pokeapi.co/api/v2/pokemon/1");
    console.log(res);
  }
  useEffect(() => {
    fetchData();
  }, []);

  return (
    <div>

    </div>
  )
}
export default App
```

Code

Selanjutnya, kita akan melakukan ekstrak data untuk name, weight, height, base_experience, dan abilities. Pertama, kita lakukan deklarasi useState dan set value untuk tiap data:

```jsx
import { useEffect, useState } from 'react'
import axios from 'axios'

function App() {
  const [name, setName] = useState('');
  const [weight, setWeight] = useState(0);
  const [height, setHeight] = useState(0);
  const [baseExperience, setBaseExperience] = useState(0);
  const [abilities, setAbilities] = useState([]);

  const fetchData = async () => {
    const res = await axios.get("https://pokeapi.co/api/v2/pokemon/1");
    setName(res.data.name);
    setWeight(res.data.weight);
    setHeight(res.data.height);
    setBaseExperience(res.data.base_experience);
    setAbilities(res.data.abilities);
  }
  useEffect(() => {
    fetchData();
  }, []);

  return (
    <div>

    </div>
  )
}
export default App
```

Code

COACHING PROGRAM Day 3
React.js

Dari code diatas, variabel name, weight, height, baseExperience, dan abilites telah terisi oleh data yang diperoleh dari API pokemon tersebut melalui fungsi setter dari hook useState diatas.

Menampilkan Data State ke Halaman Web

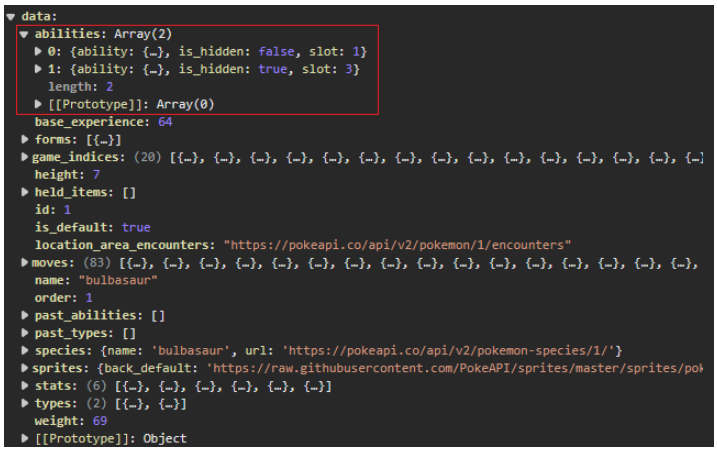
Seperti yang sudah dipelajari sebelumnya, pemanggilan variabel pada komponen dapat dilakukan dengan curly bracket seperti ini:

```
<div>
  <div>Nama: {name}</div>
  <div>Weight: {weight}</div>
  <div>Height: {height}</div>
  <div>Base Experience: {baseExperience}</div>
</div>
```

Nama: bulbasaur
Weight: 69
Height: 7
Base Experience: 64

Untuk ke empat variabel tersebut menghasilkan output yang sesuai, namun saat menambahkan baris untuk abilities, didapat error seperti ini:

Bila kalian perhatikan pada struktur data pada sub-materi Membaca Hasil dari fetching:



Dapat dilihat bahwa abilities itu bertipe data Array dengan element berupa Object, sehingga perlu dilakukan mapping data dengan fungsi map().

COACHING PROGRAM Day 3
React.js

Fungsi map() adalah metode yang ada pada objek Array di JavaScript. Ini digunakan untuk mengiterasi (melalui) semua elemen dalam array dan membuat array baru dengan hasil transformasi dari setiap elemen asli. Simplenya, fungsi map() mirip seperti for loop, namun lebih praktis karena bersifat callback, sedangkan for loop lebih ke pengendalian interasi.

```
<div>
  <div>Nama: {name}</div>
  <div>Weight: {weight}</div>
  <div>Height: {height}</div>
  <div>Base Experience: {baseExperience}</div>
  <div>Abilities:
    {abilities.map((item, index) => (
      <li key={index}>{item.ability.name}</li>
    ))}
  </div>
</div>
```

code

Dari code diatas, variabel abilities di map dengan struktur diatas, dan mengembalikan elemen berupa **<li>** dengan property **key={index}** yang artinya memberi penanda bahwa setiap elemen yang dihasilkan itu memiliki keunikan, alasan lebih lanjut baca disini. Lalu children dari <li> tersebut berupa pemanggilan **item.ability.name**, **item** merupakan callback dari fungsi map, dan bila kalian perhatikan datanya:

```
▼ abilities: Array(2)
  ▼ 0:
    ▶ ability: {name: 'overgrow', url: 'https://pokeapi.co/api/v2/ability/65/'}
      is_hidden: false
      slot: 1
    ▶ [[Prototype]]: Object
  ▼ 1:
    ▶ ability: {name: 'chlorophyll', url: 'https://pokeapi.co/api/v2/ability/34/'}
      is_hidden: true
      slot: 3
    ▶ [[Prototype]]: Object
    length: 2
  ▶ [[Prototype]]: Array(0)
```

Untuk mengakses nama dari ability, perlu masuk ke abilities > ability > name, dan karena abilities di wakilkan oleh callbal **item**, code item.ability.name akan menghasilkan nama dari setiap ability tersebut.

Nama: bulbasaur
Weight: 69
Height: 7
Base Experience: 64
Abilities:
- overgrow
- chlorophyll

Untuk eksplorasi lebih lanjut, kalian dapat mempelajari fungsi-fungsi javascript untuk mengelola object dan array seperti filter(), find(), reduce(), some(), every(), sort(), forEach(),

COACHING PROGRAM Day 3
React.js

Object.keys(), Object.values(), Object.entries(), Object.assign(), dan lain-lain. Fungsi-fungsi tersebut harus kalian pelajari dalam mengelola data dengan skala yang luas, karena framework javascript tepatnya react, pasti berurusan dengan JSON dan Object.