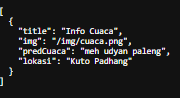
**LATIHAN**

1. **Membuat JSON HTTP Endpoints**
2. **Req Query**
   1. Bukalah aplikasi web-server anda pada visual studio code. Pastikan anda berada pada direktori **src**, lalu jalankan aplikasi menggunakan perintah **nodemon app.js -e js,hbs**
   2. Bukalah URL <http://localhost:4000/infoCuaca> pada web browser anda perhatikan bahwa ini akan menampilkan file JSON dengan format statis sebagaimana yang ada pada baris kode **app.get('/infoCuaca'………)**

****

Gambar 2. Tampilan localhost pada URL infoCuaca

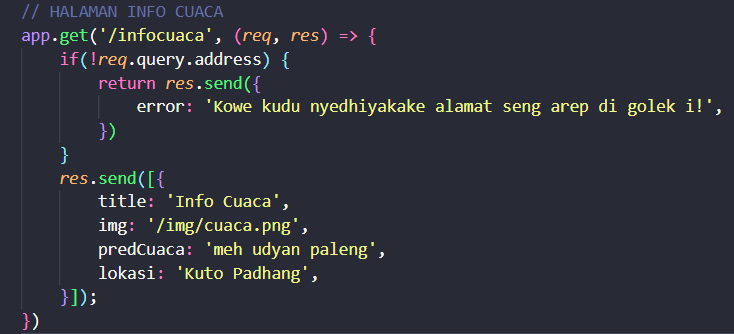
* 1. Pada tahap selanjutnya kita akan menggunakan **req.query** sebagai salah satu objek permintaan *(request)* dalam expressjs. **req.query** diisi dengan string kueri *request* yang ditemukan dalam URL. String kueri ini berbentuk pasangan kunci-nilai *(key- value)* dan dimulai setelah tanda tanya dalam URL. Jika terdapat lebih dari satu string kueri, mereka dipisahkan dengan tanda "ampersand." Contoh dapat dilihat di bawah ini.



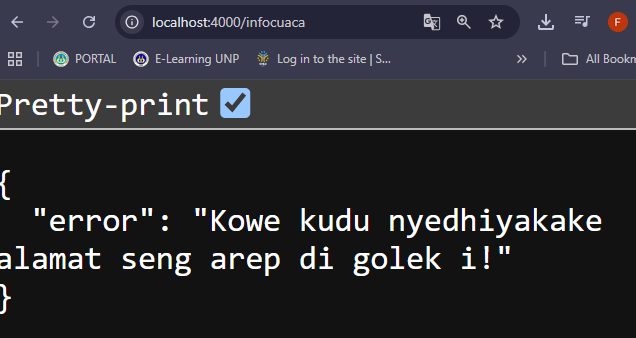
Dari kode di atas, string kueri yang digunakan adalah "nama" dan "isAuthor". Ketika permintaan ini dilakukan, objek `req.query` akan diisi dengan string kueri tersebut.

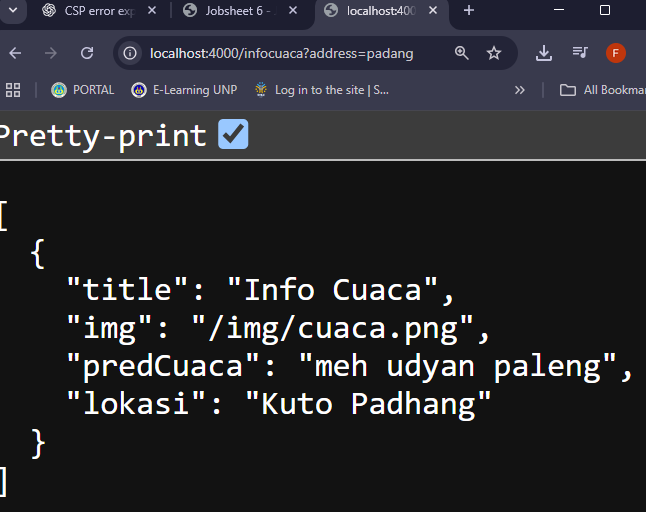
req.query = {name: "Theodore", isAuthor: true}

* 1. Cobalah ganti baris kode pada **app.get('/infocuaca'………)** dengan kode berikut ini



* 1. Cobalah akses di browser dengan mengetikan url berikut [http://localhost:4000/infoCuaca?address=padang.](http://localhost:4000/infoCuaca?address=padang)
  2. Pahami apa yang ditampilkan pada langkah 5. Lalu cobalah mengakses [http://localhost:4000/infoCuaca](http://localhost:4000/infoCuacag). Anda akan melihat bahwa aplikasi menampilkan pesan untuk memasukan lokasi yang ingin dicari. Hal ini terjadi karena anda tidak memasukan kueri **address**

****



1. **Integrasi API Wheatherstack dan Mapbox kedalam JSON HTTP Endpoints**
2. **Hentikan** nodemon anda, lalu lakukan instalasi **postman-request** dengan perintah **npm** sebagaimana yang pernah anda lakukan pada **modul 3**. Berikut adalah halamannya htt[ps://www.npmjs.com/package/postman-request](http://www.npmjs.com/package/postman-request)
3. Setelah instalasi selesai, jalankan kembali aplikasi dengan perintah **nodemon**
4. Selanjutnya, buatlah folder baru didalam folder **src** dengan nama **utils**.
5. Lalu buatlah **dua** file baru dalam folder tersebut dengan nama **geocode.js** dan prediksiCuaca.js
6. Masukan kode berikut pada file **geocode.js**

const request = require('postman-request')

const geocode = (address, callback) => {

const url ='https://api.mapbox.com/geocoding/v5/mapbox.places/'+ encodeURIComponent(address) +'.json?access\_token=API\_KEY\_DISINI'

request({ url: url, json: true}, (error, response) => { if (error){

callback('Tidak dapat terkoneksi ke layanan', undefined)

} else if (response.body.features.length === 0){

callback('Tidak dapat menemukan lokasi. Lakukan pencarian lokasi yang lain', undefined)

} else {

callback(undefined, {

latitude : response.body.features[0].center[1], longitude : response.body.features[0].center[0], location : response.body.features[0].place\_name

})

}

})

}

module.exports = geocode

**PENTING!** Jika anda menggunakan API **positionstack**, maka silakan menyesuaikan **url API** nya dan juga seluruh bagian kode yang mengandung **response.body.** Silakan akses kembali file API positionstack melalui browser anda untuk melihat data yang ingin anda akses.

1. Lalu masukan kode berikut pada file **prediksiCuaca.js**

const request = require('postman-request')

const forecast = (latitude, longitude, callback) => { const url =

['http://api.weatherstack.com/current?access\_key=API\_KEY\_DISINI&quer](http://api.weatherstack.com/current?access_key=API_KEY_DISINI&quer) y='+encodeURIComponent(latitude)+','+encodeURIComponent(longitude)+ '&units=m';

request({ url: url, json: true}, (error, response) => { if(error) {

callback('Tidak dapat terkoneksi ke layanan',

undefined)

} else if(response.body.error) {

callback('Tidak dapat menemukan lokasi',undefined)

} else {

callback(undefined, 'Info Cuaca: ' +

response.body.current.weather\_descriptions[0] +'. '+

'Suhu saat ini adalah ' + response.body.current.temperature + ' derajat. ' +

'Index UV adalah ' + response.body.current.uv\_index + ' nm. ' +

'Visibilitas ' + response.body.current.visibility +

' kilometer'

)

}

})

}

module.exports = forecast

1. Selanjutnya, ubahlah baris kode **app.get('/infocuaca'………)** pada file **app.js** yang ada di folder **src** dengan kode berikut ini

app.get('/infocuaca', (req, res) => { if(!req.query.address){

return res.send({

error:'Kamu harus memasukan lokasi yang ingin dicari'

})

}

geocode(req.query.address, (error, { latitude, longitude, location } = {}) => {

if (error){

return res.send({error})

}

forecast(latitude, longitude, (error, dataPrediksi) => { if (error){

return res.send({error})

}

res.send({

prediksiCuaca: dataPrediksi, lokasi: location,

address: req.query.address

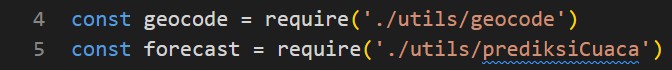
})

})

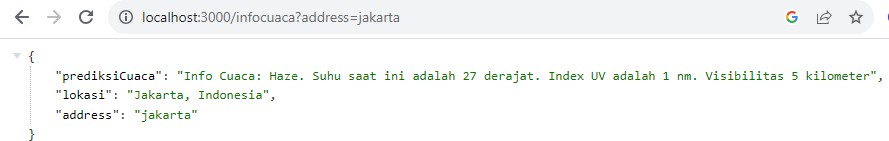
})

})

1. Perlu diperhatikan juga bahwa anda harus meng-*import* file **geocode** dan

**prediksiCuaca** dengan menambahkan baris kode berikut pada awal kode app.js

Gambar 3. Variabel geocode dan forecast untuk import file geocode dan prediksiCuaca

1. Cobalah mengakses url <http://localhost:3000/infocuaca?address=jakarta> berikut dibrowser anda. Cobalah mengganti kata ‘jakarta’ dengan keyword lainnya. Perhatikanlah bahwa objek **prediksiCuaca**, **lokasi**, dan **address** sekarang bersifat dinamis, bukanlah lagi statis

Gambar 4. Akses JSON HTTP Endpoints yang bersifat dinamis

1. **Menampilkan data API menggunakan Method Fetch**

Pada bagian ini, anda akan membuat form pencarian lokasi agar anda dapat mengecek cuaca berdasarkan lokasi yang dicari. Anda akan menggunakan method **fetch.** Di Node.js, fetch bukanlah sebuah method bawaan *(built-in method),* tetapi biasanya digunakan dalam lingkungan browser, bukan pada *runtime* Node.js. **fetch** adalah API JavaScript yang digunakan untuk mengambil *(fetch)* data dari sumber eksternal seperti server web atau API REST. API ini sangat umum digunakan di lingkungan browser untuk membuat permintaan HTTP.

1. Bukalah file **public/js/app.js**, kemudian masukanlah kode berikut ini

const weatherform = document.querySelector('form') const search = document.querySelector('input') const pesanSatu = document.querySelector('#pesan-1') const pesanDua = document.querySelector('#pesan-2')

// pesanSatu.textContent = 'From javascript' weatherform.addEventListener('submit', (e) => {

e.preventDefault()

const location = search.value

pesanSatu.textContent = 'Sedang mencari lokasi ..' pesanDua.textContent = ''

fetch(['http://localhost:3000/infocuaca?address='+](http://localhost:3000/infocuaca?address=%27%2B) location).then((response)=>{

response.json().then((data)=>{ if(data.error){

pesanSatu.textContent = data.error

} else {

pesanSatu.textContent = data.lokasi pesanDua.textContent = data.prediksiCuaca

}

})

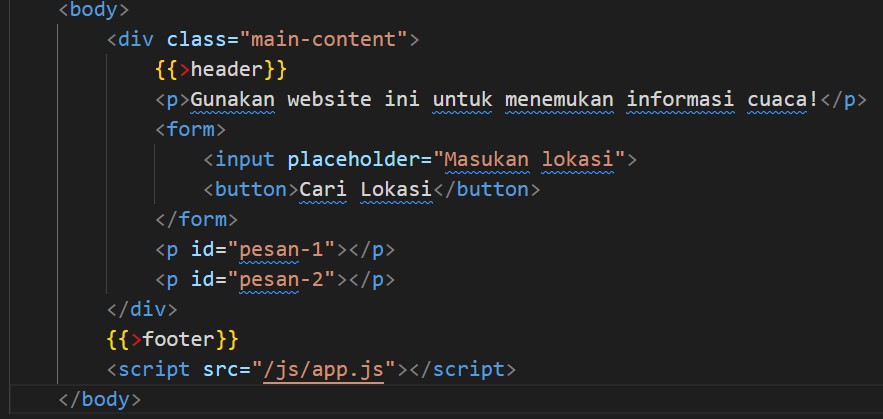
})

})

# Catatan:

* 1. **querySelector** diatas adalah sebuah metode yang digunakan di lingkungan browser pada JavaScript untuk mengakses elemen-elemen HTML di halaman web dengan menggunakan selector CSS.
  2. Perintah **fectch(…)** diatas digunakan untuk mengambil data dari JSON HTTP Endpoints yang telah kita buat sebelumnya. Jadi, pada dasarnya kita akan menggunakan data API yang telah kita buat sendiri untuk kemudian kita tampilkan pengguna

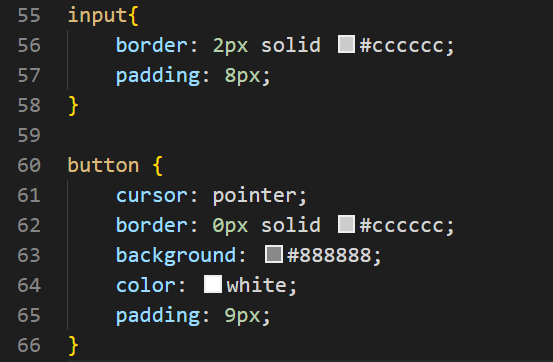
1. Lalu ubahlah baris kode pada bagian **<body>** pada file **index.hbs** dengan kode berikut ini



Gambar 5. Kode form pencarian pada file index.hbs

# PENTING!

* 1. Perhatikan bahwa javascript **/js/app.js** diletakan pada bagian akhir, bukan lagi berada pada bagian **<head>** seperti sebelumnya. Hal ini bertujuan agar file tersebut di eksekusi setelah proses pencarian pada form dilakukan
  2. Hapuslah baris script ini dari bagian **<head>** di semua file .hbs diantaranya **bantuan.hbs, tentang.hbs** dan **404.hbs**. Hal ini dilakukan karena file ini hanya dibutuhkan pada **index.hbs** saja.

1. ****Tambahkanlah kode berikut ini pada file **styles.css**

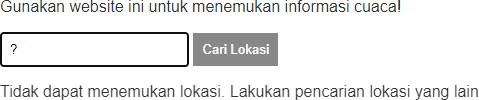
Gambar 6. Kode style untuk form input dan button

1. Silakan akses aplikasi cek cuaca anda pada browser. Lakukan pencarian dengan mengetikan lokasi atau tempat yang ingin dicari, misalnya: ‘**Jakarta**’atau ‘**Universitas Negeri Padang’**. Pastikan bahwa tampilannya adalah seperti berikut ini



Gambar 7. Tampilan akhir aplikasi cek cuaca apabila pencarian ditemukan

1. Coba juga untuk mengetik tanda **?** pada kotak pencarian dan pastikan bahwa yang tampil adalah seperti gambar berikut ini



Gambar 8. Tampilan akhir aplikasi cek cuaca apabila pencarian tidak ditemukan

**TUGAS**

1. Carilah template HTML yang menarik di google ataupun platform lainnya, kemudian gantilah tampilan pada halaman **tentang saya.** Anda dapat mengubah tampilan ini dengan memodifikasi file **styles.css** (jika diperlukan) dan yang terutama modifikasi file **tentang.hbs**. Sesuaikan tampilan ini dengan keinginan anda. Tujuan halaman ini adalah menampilkan informasi tentang aplikasi dan pengembang aplikasi
2. Ubah juga tampilan pada halaman **bantuan** dengan memodifikasi file **bantuan.hbs.** Tujuan halaman ini adalah sebagai **FAQ (Frequently Asked Questions**) yang berisi list pertanyaan tentang aplikasi yang kita kembangkan