
NodeJS Dasar

DNS



DNS

- DNS merupakan standard library yang bisa digunakan untuk bekerja dengan DNS (domain name server)
- <https://nodejs.org/docs/latest-v22.x/api/dns.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/dns.html>

Kode : DNS



```
dns.mjs ×
1 import dns from "dns";
2
3 function callback(error, ip) {
4     console.info(ip);
5 }
6
7 dns.lookup( hostname: "www.programmerzamannow.com", callback)
8
```



Kode : DNS Promise

```
JS dns.mjs  ×  
JS dns.mjs > ...  
1 import dns from "dns/promises";  
2  
3 const address = await dns.lookup("www.umy.ac.id");  
4  
5 console.info(address.address);  
6 console.info(address.family);  
7
```

```
1 import dns from "dns/promises";
2
3 const address = await dns.lookup("www.umy.ac.id");
4
5 console.info(address.address);
6 console.info(address.family);
```

js

```
1 | import dns from "dns/promises";
```

- ◆ Baris ini mengimpor modul dns/promises dari Node.js, yang menyediakan API berbasis *Promise* untuk fungsi DNS.
- ◆ Artinya, Anda bisa menggunakan `await` untuk operasi DNS seperti `lookup`, yang sebelumnya hanya tersedia via callback.

js

```
3 | const address = await dns.lookup("www.umy.ac.id");
```

- ◆ Di baris ini, dilakukan pencarian alamat IP dari nama domain `www.umy.ac.id` menggunakan `dns.lookup`.
- ◆ Karena menggunakan `await`, maka kode ini **menunggu** hingga proses DNS selesai dan hasilnya disimpan ke dalam variabel `address`.

📌 **Hasilnya** berupa objek:

js

```
{
  address: '103.109.54.19', // contoh hasil IP
  family: 4 // 4 = IPv4, 6 = IPv6
}
```

Events



Events

- Events adalah standard library di NodeJS yang bisa digunakan sebagai implementasi Event Listener
- Di dalam Events, terdapat sebuah class bernama EventEmitter yang bisa digunakan untuk menampung data listener per jenis event.
- Lalu kita bisa melakukan emmit untuk mentrigger jenis event dan mengirim data ke event tersebut
- <https://nodejs.org/docs/latest-v22.x/api/events.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/events.html>

Kode : Events

JS events.mjs X

```
JS events.mjs > ...
1 import {EventEmitter} from "events";
2
3 const emitter = new EventEmitter();
4
5 emitter.addListener("hello", (name) => {
6   |   console.info(`Hello ${name}`);
7 })
8 emitter.addListener("hello", (name) => {
9   |   console.info(`Halo ${name}`);
10 })
11
12 emitter.emit("hello", "Asroni");
```

Kode:

js

```
1 | import { EventEmitter } from "events";
```

- ◆ Mengimpor EventEmitter dari modul bawaan Node.js (events).
- ◆ EventEmitter adalah class yang memungkinkan objek JavaScript untuk memancarkan (*emit*) dan menangani (*listen to*) event.

js

```
3 | const emitter = new EventEmitter();
```

- ◆ Membuat instance emitter dari EventEmitter.
- ◆ Objek emitter ini akan digunakan untuk *menambahkan listener* dan *memicu* event.

js

```
5 | emitter.addListener("hello", (name) => {  
6 |   console.info(`Hello ${name}`);  
7 | })
```

- ◆ Menambahkan listener pertama untuk event "hello".
- ◆ Fungsi ini akan dipanggil saat event "hello" dipicu (*emit*), dan akan mencetak:

Hello Asroni

js

```
8 | emitter.addListener("hello", (name) => {  
9 |   console.info(`Halo ${name}`);  
10| })
```

- ◆ Menambahkan listener kedua untuk event "hello".
- ◆ Fungsi ini juga akan dijalankan saat "hello" dipanggil, mencetak:

Halo Asroni

js

```
12| emitter.emit("hello", "Asroni");
```

- ◆ Memicu (emit) event "hello" dan mengirim argumen "Asroni" ke semua listener yang terdaftar pada event tersebut.
- ◆ Kedua listener akan dieksekusi secara berurutan.

Output di Konsol:

Hello Asroni
Halo Asroni

Globals

Globals

- Di dalam NodeJS, terdapat library berupa variable atau function yang secara global bisa diakses dimana saja, tanpa harus melakukan import
- Kita bisa melihat detail apa saja fitur yang terdapat secara global di halaman dokumentasinya
- <https://nodejs.org/docs/latest-v22.x/api/globals.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/globals.html>

Kode : Globals

```
JS globals.mjs > ...  
1  
2   setTimeout(() => {  
3     |   console.info("Hello Globals");  
4   }, 2000);
```

```
js  
Copy  
setTimeout(() => {
```

Baris ini menggunakan fungsi global `setTimeout`, yang berfungsi untuk menunda eksekusi fungsi tertentu dalam waktu tertentu; dalam hal ini, menggunakan fungsi panah kosong `() => { ... }` sebagai callback yang akan dijalankan setelah penundaan.

```
js  
Copy  
console.info("Hello Globals");
```

Baris ini adalah isi dari fungsi callback, yang akan mencetak teks "Hello Globals" ke konsol saat fungsi ini dijalankan.

```
js  
Copy  
, 2000);
```

Baris ini menutup fungsi `setTimeout` dan memberikan argumen kedua berupa `2000` (dalam milidetik), artinya kode dalam fungsi callback akan dijalankan **setelah 2 detik** sejak program dimulai.

Process



Process

- Process merupakan standard library yang digunakan untuk mendapatkan informasi proses NodeJS yang sedang berjalan
- Process juga merupakan instance dari EventEmitter, sehingga kita bisa menambahkan listener kedalam Process
- <https://nodejs.org/docs/latest-v22.x/api/process.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/process.html>

Kode : Process



A screenshot of a code editor showing a file named `process.mjs`. The code is written in JavaScript and uses the `process` module. It includes an `exit` event listener that logs the exit code and the Node.js version. It also prints the command-line arguments, report, and environment variables. Finally, it exits with a code of 1 and logs a message indicating it won't print because it has already exited.

```
process.mjs
1 import process from "process";
2
3 process.addListener( event: "exit", [ listener: function (exitCode: number) {
4     console.info( data: `NodeJS exit with code ${exitCode}`);
5 });
6
7 console.info(process.version);
8 console.table(process.argv);
9 console.table(process.report);
10 console.table(process.env)
11
12 process.exit( code: 1);
13
14 console.info( data: "Not Printed because already exit");
15
```

```
js
```

 Copy

```
import process from "process";
```

Baris ini mengimpor modul bawaan Node.js bernama `process`, yang menyediakan informasi dan kontrol tentang proses Node.js yang sedang berjalan.

```
js
```

 Copy

```
process.addListener("exit", (exitCode) => {
```

Baris ini mendaftarkan event listener pada event `"exit"` dari objek `process`, yang akan dipanggil saat proses Node.js akan keluar.

```
js
```

 Copy

```
console.info(`NodeJS exit with code ${exitCode}`);
```

Di dalam fungsi callback, baris ini mencetak ke konsol informasi bahwa Node.js keluar dan menampilkan kode keluar (`exit code`) yang dikembalikan oleh proses.

```
js
```

 Copy

```
})
```

Baris ini menutup definisi fungsi listener tersebut.

```
js
```

 Copy

```
console.info(process.version);
```

Baris ini mencetak versi Node.js yang sedang digunakan.

```
js
```

 Copy

```
console.table(process.argv);
```

Baris ini menampilkan argumen baris perintah (`command line arguments`) yang diteruskan saat menjalankan file, dalam bentuk tabel.

```
js
```

 Copy

```
console.table(process.report);
```

Baris ini menampilkan laporan internal proses dalam bentuk tabel, yang bisa berisi informasi debugging.

```
js
```

 Copy

```
console.table(process.env);
```

Baris ini mencetak daftar variabel lingkungan (`environment variables`) dalam bentuk tabel.

```
js
```

 Copy

```
process.exit(1);
```

Baris ini menghentikan proses Node.js secara eksplisit dan mengembalikan kode keluar 1, yang berarti proses berakhir dengan kegagalan atau kondisi tertentu.

```
js
```

 Copy

```
console.info("Hello");
```

Baris ini tidak akan pernah dijalankan karena baris sebelumnya `process.exit(1)`; telah mengakhiri proses sebelum mencapai baris ini.

Readline



Readline

- Readline merupakan standard library yang digunakan untuk membaca input
- Namun pada saat dibuat video ini, Readline hanya mendukung versi callback di versi NodeJS LTS 16.
- Di NodeJS 17 sudah mendukung Promise sehingga lebih mudah digunakan, namun itu pun masih dalam tahap experimental
- <https://nodejs.org/docs/latest-v22.x/api/readline.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/readline.html>

Kode : Readline



```
readline.mjs ×
JS
1 import process from "process";
2 import readline from "readline";
3
4 const input = readline.createInterface({ options: {
5   input: process.stdin,
6   output: process.stdout
7 }});
8
9 input.question("Siapa nama anda? : ", callback: function (nama: string) {
10   console.info(`Hello ${nama}`);
11   input.close();
12 })
13
```

```
js
```

```
import readline from "readline";
```

Baris ini mengimpor modul `readline` bawaan Node.js yang digunakan untuk membaca input dari pengguna lewat terminal baris per baris.

```
js
```

```
import process from "process";
```

Baris ini mengimpor modul `process` untuk mengakses `stdin` (input) dan `stdout` (output) dari proses saat ini.

```
js
```

```
const input = readline.createInterface({
```

Baris ini membuat antarmuka input/output menggunakan fungsi `createInterface` dari modul `readline`.

```
js
```

```
  input: process.stdin,
```

Baris ini mengatur bahwa data input akan diambil dari `stdin`, yaitu terminal pengguna.

```
js
```

```
  output: process.stdout,
```

Baris ini mengatur bahwa output akan ditampilkan di terminal (`stdout`).

```
js
```

Copy

```
});
```

Baris ini menutup konfigurasi `createInterface`.

```
js
```

Copy

```
  input.question("Siapa nama Anda? ", (name) => {
```

Baris ini menampilkan pertanyaan "Siapa nama Anda?" ke pengguna dan menunggu input, lalu menjalankan fungsi callback dengan argumen `name` berisi jawaban pengguna.

```
js
```

Copy

```
    console.info(`Halo ${name}`);
```

Baris ini mencetak sapaan ke layar dengan nama yang dimasukkan pengguna, misalnya: Halo Asroni.

```
js
```

Copy

```
  input.close();
```

Baris ini menutup antarmuka `readline` setelah input selesai diproses.

```
js
```

Copy

```
});
```

Baris ini menutup fungsi callback untuk pertanyaan `question`.

Report

Report

- Report merupakan fitur yang terdapat di NodeJS untuk membuat laporan secara otomatis dalam file ketika sesuatu terjadi pada aplikasi NodeJS kita
- <https://nodejs.org/docs/latest-v22.x/api/report.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/report.html>



Kode : Error pada Aplikasi NodeJS



```
report.mjs
1 import process from "process";
2
3 process.report.reportOnFatalError = true;
4 process.report.reportOnUncaughtException = true;
5 process.report.reportOnSignal = true;
6 process.report.filename = "report.json";
7
8 function sampleError() {
9     throw new Error("Ups");
10}
11
12 sampleError();
```

```
js
```

```
import process from "process";
```

Baris ini mengimpor modul process, yang menyediakan informasi dan kontrol terhadap proses Node.js yang sedang berjalan.

```
js
```

```
process.report.reportOnFatalError = true;
```

Baris ini mengaktifkan pelaporan otomatis ketika terjadi kesalahan fatal, seperti segmentasi memori.

```
js
```

```
process.report.reportOnSignal = true;
```

Baris ini mengatur agar laporan juga dibuat saat proses menerima sinyal sistem (misalnya SIGTERM).

```
js
```

```
process.report.reportOnUncaughtException = true;
```

Baris ini memastikan bahwa jika terjadi pengecualian (error) yang tidak tertangani, Node.js akan menghasilkan laporan diagnostik.

```
js
```

```
process.report.filename = "report.json";
```

Baris ini menentukan bahwa file hasil laporan error akan disimpan dengan nama report.json.

```
js
```

```
function sampleError() {
```

Baris ini mendefinisikan fungsi sampleError yang akan digunakan untuk mensimulasikan error.

```
js
```

```
throw new Error("Ups");
```

Baris ini melempar sebuah objek error baru dengan pesan "Ups", yang akan memicu mekanisme pelaporan error tadi.

```
js
```

```
}
```

Menutup definisi fungsi.

```
js
```

```
sampleError();
```

Baris ini memanggil fungsi sampleError sehingga error akan terjadi, lalu sistem akan secara otomatis menghasilkan file report.json yang berisi informasi lengkap tentang error tersebut.

Buffer



Buffer

- Buffer merupakan object yang berisikan urutan byte dengan panjang tetap.
- Buffer merupakan turunan dari tipe data Uint8Array
- <https://nodejs.org/docs/latest-v22.x/api/buffer.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/buffer.html>

Kode : Buffer

```
js buffer.mjs > ...
1
2 const buffer = Buffer.from("Asroni Sukirman");
3
4 console.info(buffer); // Tampilkan buffer dalam bentuk byte
5 console.info(buffer.toString()); // Tampilkan buffer sebagai string
6
7 const reversed = Buffer.from([...buffer].reverse()); // Balik isi buffer
8 console.info(reversed.toString()); // Tampilkan string yang sudah dibalik
```

◆ Baris 2:

js

```
const buffer = Buffer.from("Asroni Sukirman");
```

- Membuat sebuah **Buffer** dari string "Asroni Sukirman".
- **Buffer** adalah representasi biner (byte array) dari data string.
- Misalnya: "A" menjadi 0x41, "s" menjadi 0x73, dst.

◆ Baris 4:

js

```
console.info(buffer);
```

- Menampilkan isi **buffer** dalam bentuk array byte hexadecimal.
- Contoh output:

bash

```
<Buffer 41 73 72 6f 6e 69 20 53 75 6b 69 72 6d 61 6e>
```

Buffer Encoding

- Buffer juga bisa digunakan untuk melakukan encoding dari satu encoding ke encoding yang lain
- Ada banyak encoding yang didukung oleh Buffer, misal utf8, ascii, hex, base64, base64url dan lain-lain

```
JS buffer-encoding.mjs ●

JS buffer-encoding.mjs > ...
1 import { clear } from "console";
2
3 const buffer = Buffer.from("Asroni Sukirman", "utf8");
4
5 console.info(buffer.toString());
6 console.info(buffer.toString("hex"));
7 console.info(buffer.toString("base64"));
8
9 const bufferBase64 = Buffer.from("QXNyb25pIFN1a2lybWFu", "base64");
10 console.info(bufferBase64.toString("utf8"));
11 |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

- asroni@asronis-MacBook-Air belajar-nodejs-dasar-main % node buffer-encoding.mjs
Asroni Sukirman
4173726f6e692053756b69726d616e
QXNyb25pIFN1a2lybWFu
Asroni Sukirman
- asroni@asronis-MacBook-Air belajar-nodejs-dasar-main % []

Kode : Buffer Encoding

```
buffer-encoding.mjs ×

1 const buffer = Buffer.from(str: "Eko Kurniawan Khannedy", encoding: "utf8");
2
3 console.info(buffer.toString(encoding: "base64"));
4 console.info(buffer.toString(encoding: "hex"));
5
6
7 const buffer2 = Buffer.from(str: "RWtvIEt1cm5pYXdhbiBLaGFubmVkeQ==", encoding: "base64url");
8
9 console.info(buffer2.toString(encoding: "utf8"));

|
```

Stream



Stream

- Stream adalah standard library untuk kontrak aliran data di NodeJS
- Ada banyak sekali Stream object di NodeJS
- Stream bisa jadi object yang bisa dibaca, atau bisa di tulis, dan Stream adalah turunan dari EventEmitter
- <https://nodejs.org/docs/latest-v22.x/api/stream.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/stream.html>



Kode : Stream

```
JS stream.mjs X

JS stream.mjs > ...
1 import fs from "fs";
2
3 const writer = fs.createWriteStream("target.log");
4
5 writer.write("Asroni\n");
6 writer.write("Sukirman\n");
7 writer.write("Kalipin\n");
8 writer.end();
9
10 const reader = fs.createReadStream("target.log");
11 reader.addListener("data", (data) => {
12   console.info(data.toString());
13 });


```

Timer



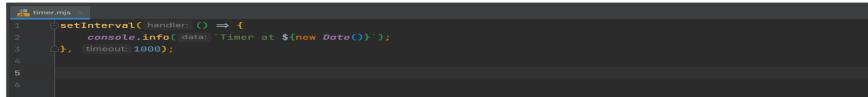
Timer

- Timer merupakan standard library untuk melakukan scheduling
- Function di Timer terdapat di globals, sehingga kita bisa menggunakannya tanpa melakukan import, namun semua function Timer menggunakan Callback
- Jika kita ingin menggunakan Timer versi Promise, kita bisa meng-import dari module timer/promise
- <https://nodejs.org/docs/latest-v22.x/api/timers.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/timers.html>

Kode : Timer

```
JS timer.mjs > ...
```

```
1
2  setInterval(() => {
3    |   console.info(`Start time at ${new Date()}`);
4  }, 1000);
|
```



Kode : Timer Promise

```
JS timer-promise.mjs > ...
1
2 import timers from "timers/promises";
3
4 console.info(new Date());
5 const name = await timers.setTimeout(5000, "Jeda");
6 console.info(new Date());
7 console.info(name);
8
9 for await (const startTime of timers.setInterval(1000, "ignored")){
10   console.info(`Start time at ${new Date()}`);
11 }
```



Net



Net

- Net merupakan standard library yang bisa digunakan untuk membuat network client dan server berbasis TCP
- Net Server dan Client merupakan object Stream, sehingga kita bisa baca datanya, tulis datanya dan juga menambahkan listener
- <https://nodejs.org/docs/latest-v22.x/api/net.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/net.html>

Kode : Net Server

```
JS net-server.mjs ×

JS net-server.mjs > ...

1 import net from "net";
2
3 const server = net.createServer((client) => {
4     console.info("Client connected");
5
6     client.addListener("data", (data) => {
7         console.info(`Receive data ${data.toString()}`);
8         client.write(`Hello ${data.toString()}\r\n`);
9     }
10
11 });
12
13 server.listen(3000, "localhost");
```

Kode : Net Client

```
JS net-client.mjs × JS net-server.mjs

JS net-client.mjs > ...
1 import net from "net";
2
3 const client = net.createConnection({
4     port: 3000,
5     host: "localhost"
6 });
7
8 client.addListener("data", (data) => {
9     console.info(`Receive data from server : ${data.toString()}`)
10 });
11
12 setInterval(() => {
13     client.write("Asroni\r\n");
14     //client.write(`${process.argv[2]}\r\n`);
15 }, 2000);
16
17
```

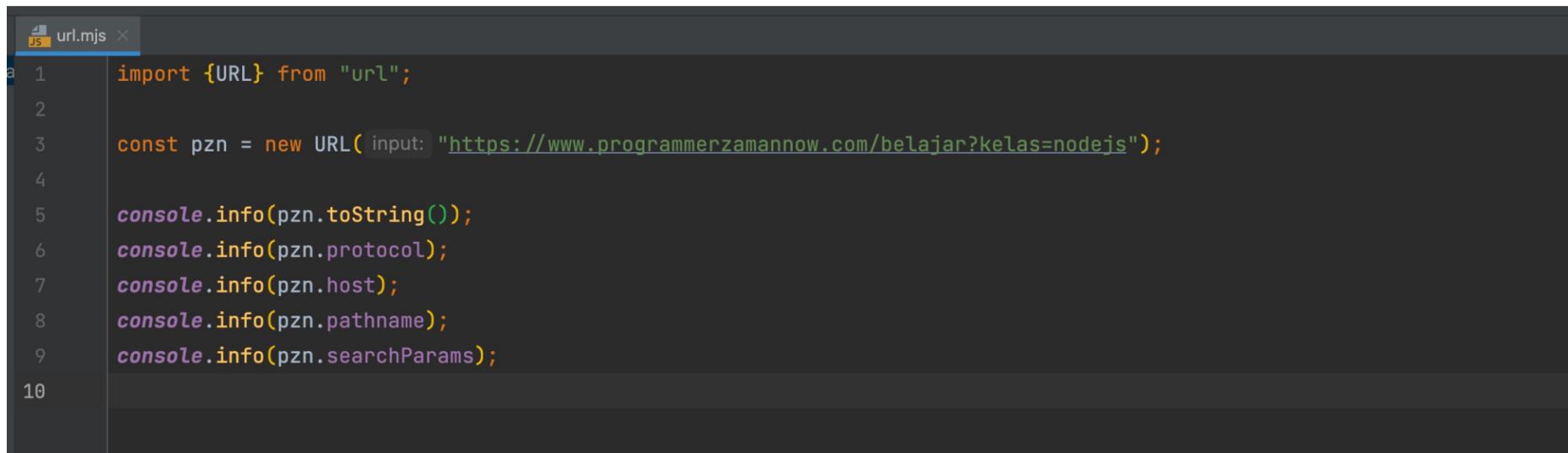
URL



URL

- URL merupakan standard library untuk bekerja dengan URL
- <https://nodejs.org/docs/latest-v22.x/api/url.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/url.html>

Kode : URL



A screenshot of a code editor showing a file named "url.mjs". The code uses the "url" module to parse a URL and extract its components. The code is numbered from 1 to 10.

```
1 import {URL} from "url";
2
3 const pzn = new URL("https://www.programmerzamannow.com/belajar?kelas=nodejs");
4
5 console.info(pzn.toString());
6 console.info(pzn.protocol);
7 console.info(pzn.host);
8 console.info(pzn.pathname);
9 console.info(pzn.searchParams);
10
```

Kode : Mengubah URL



```
url.mjs
1 import {URL} from "url";
2
3 const pzn = new URL(input: "https://www.programmerzamannow.com/belajar?kelas=nodejs");
4
5 pzn.host = "www.khannedy.com";
6 const searchParam = pzn.searchParams;
7 searchParam.append(name: "status", value: "premium");
8
9 console.info(pzn.toString());
10
```

```
JS url.mjs > ...
1 import {URL} from "url";
2
3 const pzn = new URL("https://sport.detik.com/sepakbola/jadwal");
4
5 pzn.host = "www.sportaja.com";
6
7 pzn.searchParams.append("status", "premium");
8
9 console.info(pzn.toString());
10 console.info(pzn.href);
11 console.info(pzn.protocol);
12 console.info(pzn.host);
13 console.info(pzn.pathname);
14 console.info(pzn.searchParams);
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

- asroni@asronis-MacBook-Air belajar-nodejs-dasar-main % node url.mjs
https://www.sportaja.com/sepakbola/jadwal?status=premium
https://www.sportaja.com/sepakbola/jadwal?status=premium
https:
www.sportaja.com
/sepakbola/jadwal
URLSearchParams { 'status' => 'premium' }

Util



Util

- Util adalah standard library yang berisikan utility-utility yang bisa kita gunakan untuk mempermudah pembuatan kode program di NodeJS
- <https://nodejs.org/docs/latest-v22.x/api/util.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/util.html>

Kode : Util



```
util.mjs ×
1 import util from "util";
2
3 console.info(util.format(`Nama : %s`, "Eko"));
4
5 const person = {firstName: "Eko", lastName: "Khannedy"};
6 console.info(util.format("Person : %j", person));
7
```

Zlib

Zlib

- Zlib adalah standard library yang digunakan untuk melakukan kompresi menggunakan Gzip
- <https://nodejs.org/dist/latest-v16.x/docs/api/zlib.html>

Kode : Zlib Compress



```
JS zlib.mjs ×
1 import zlib from "zlib";
2 import fs from "fs";
3
4 const source = fs.readFileSync( path: "zlib.mjs");
5 const result = zlib.gzipSync(source);
6
7 fs.writeFileSync( file: "zlib.mjs.txt", result);
```

Kode : Zlib Decompress



```
JS zlib-decompress.mjs ×
1 import zlib from "zlib";
2 import fs from "fs";
3
4 const source = fs.readFileSync( path: "zlib.mjs.txt");
5 const result = zlib.unzipSync(source);
6 console.info(result.toString());
7
8
```