



NodeJS Dasar



Agenda

- Pengenalan NodeJS
- Pengenalan Concurrency
- NodeJS Architecture
- Menginstall NodeJS
- NodeJS REPL
- Standard Library
- Dan lain-lain

Zlib



Zlib

- Zlib adalah standard library yang digunakan untuk melakukan kompresi menggunakan Gzip
- <https://nodejs.org/docs/latest-v22.x/api/zlib.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/zlib.html>



Kode : Zlib Compress

```
JS  zlib.mjs x
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);
8
```



Kode : Zlib Decompress

zlib-decompres.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
```

belajar-nodejs-dasar-main > JS zlib-decompress.mjs > ...

```
1 import fs from "fs";
2 import zlib from "zlib";
3
4 const source = fs.readFileSync("zlib-compress.mjs.txt");
5 console.info(source.toString());
6
7 const result = zlib.unzipSync(source);
8 console.info(result.toString());
9
```

PROBLEMS


OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

POSTMAN CONSOLE

 zsh - b

● asroni@asronis-MacBook-Air belajar-nodejs-dasar-main % node zlib-decompress.mjs

}0A0 E00b^M0

0n00'P0(dfk0oJYw0007\

+D00%00b00n0000zcB0EA00a0(000Q00oC0]CcQ0000w0Q00000{oT0r0FoL0eR<09]0^zjw000Gn0

```
import fs from "fs";
import zlib from "zlib";
```

```
const source = fs.readFileSync("zlib-compress.mjs");
const result = zlib.gzipSync(source);
```

```
fs.writeFileSync("zlib-compress.mjs.txt", result);
```

○ asroni@asronis-MacBook-Air belajar-nodejs-dasar-main % █

Console



Console

- Console adalah standard library yang sudah sering kita gunakan
- Secara global, object console bisa kita gunakan tanpa harus melakukan import module, dan console melakukan print text nya ke stdout
- Namun jika kita juga bisa membuat object Console sendiri jika kita mau
- <https://nodejs.org/docs/latest-v22.x/api/console.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/console.html>



Kode : Console

```
console.mjs x
1 import {Console} from "console";
2 import fs from "fs";
3
4 const logFile = fs.createWriteStream( path: "application.log");
5
6 const log = new Console({
7   stdout: logFile,
8   stderr: logFile
9 });
10
11 log.info("Hello world")
12 log.error("Ups");
13
```

≡ application.log ×

□ ...

belajar-nodejs-dasar-main > ≡ application.log

```
1 Hello World
2 Hello World
```



(index)	Values
firstName	'Asroni'
lastName	'Sukirman'

JS console.mjs ×

belajar-nodejs-dasar-main > JS console.mjs > ...

```
1
2 import {Console} from "console";
3 import fs from "fs";
4
5 const file = fs.createWriteStream("application.log");
6
7 const log = new Console({
8   stdout: file,
9   stderr: file,
10 })
11
12 log.info("Hello World");
13 log.error("Hello World");
14
15 const person = {
16   firstName: "Asroni",
17   lastName: "Sukirman",
18 }
19
20 log.table(person);
21
```

Worker Threads



Worker Threads

- Worker Threads adalah standard library yang bisa kita gunakan untuk menggunakan thread ketika mengeksekusi JavaScript secara paralel
- Worker Threads sangat cocok ketika kita membuat kode program yang butuh jalan secara paralel, dan biasanya kasusnya adalah ketika kode program kita membutuhkan proses yang CPU intensive, seperti misalnya enkripsi atau kompresi
- Cara kerja Worker Threads mirip dengan Web Worker di JavaScript Web API
- https://nodejs.org/docs/latest-v22.x/api/worker_threads.html
- https://nodejs.org/dist/latest-v16.x/docs/api/worker_threads.html



Kode : Main Thread

```
worker-main.mjs x
1  import {threadId, Worker} from "worker_threads";
2
3  const worker1 = new Worker( filename: "./worker.mjs");
4  const worker2 = new Worker( filename: "./worker.mjs");
5
6  worker1.addListener( event: "message", listener: function (message) {
7      console.info( data: `thread-${threadId} receive message : ${message}`);
8  });
9  worker2.addListener( event: "message", listener: function (message) {
10     console.info( data: `thread-${threadId} receive message : ${message}`);
11 });
12
13 worker1.postMessage( value: 10);
14 worker2.postMessage( value: 10);
15
```



Kode : Worker Thread

```
worker.mjs x
1  import {parentPort, threadId} from "worker_threads";
2
3  parentPort.addListener( event: "message", listener: function (message) {
4    for (let i = 0; i < message; i++) {
5      console.info( data: `thread-${threadId} send message ${i}`);
6      parentPort.postMessage(i);
7    }
8    parentPort.close();
9  })
```

10

JS worker-main.mjs X

belajar-nodejs-dasar-main > JS worker-main.mjs > ...

```
1 import {threadId, Worker} from "worker_threads";
2
3 const worker1 = new Worker("./worker.mjs");
4 const worker2 = new Worker("./worker.mjs");
5
6 worker1.addListener("message", (message) => {
7   console.info(`Thread-${threadId} receive from worker 1 : ${message}`);
8 })
9 worker2.addListener("message", (message) => {
10   console.info(`Thread-${threadId} receive from worker 2 : ${message}`);
11 })
12
13 worker1.postMessage(10);
14 worker2.postMessage(10);
15
```

JS worker.mjs X

belajar-nodejs-dasar-main > JS worker.mjs > ...

```
1 import {threadId, parentPort} from "worker_threads";
2
3 parentPort.addListener("message", (message) => {
4   for (let i = 0; i < message; i++) {
5     console.info(`Thread-${threadId} send message ${i}`);
6     parentPort.postMessage(i);
7   }
8   parentPort.close();
9 })
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

zsh - belajar-nodejs-dasar-m

asroni@asronis-MacBook-Air belajar-nodejs-dasar-main % node worker-main.mjs

```
Thread-1 send message 0
Thread-0 receive from worker 1 : 0
Thread-0 receive from worker 1 : 1
Thread-0 receive from worker 1 : 2
Thread-0 receive from worker 1 : 3
Thread-0 receive from worker 1 : 4
Thread-0 receive from worker 1 : 5
Thread-0 receive from worker 1 : 6
Thread-0 receive from worker 1 : 7
Thread-0 receive from worker 1 : 8
Thread-0 receive from worker 1 : 9
Thread-2 send message 0
Thread-0 receive from worker 2 : 0
Thread-0 receive from worker 2 : 1
Thread-0 receive from worker 2 : 2
Thread-0 receive from worker 2 : 3
Thread-0 receive from worker 2 : 4
Thread-0 receive from worker 2 : 5
Thread-0 receive from worker 2 : 6
Thread-0 receive from worker 2 : 7
Thread-0 receive from worker 2 : 8
Thread-0 receive from worker 2 : 9
Thread-1 send message 1
Thread-1 send message 2
Thread-1 send message 3
Thread-1 send message 4
Thread-1 send message 5
Thread-1 send message 6
Thread-1 send message 7
Thread-1 send message 8
Thread-1 send message 9
Thread-2 send message 1
Thread-2 send message 2
Thread-2 send message 3
```

HTTP Client

<https://css-tricks.com/hookbin-capture-inspect-http-requests/>



HTTP Client

- NodeJS juga memiliki standard library untuk HTTP
- Salah satu fitur di module HTTP adalah HTTP Client, dimana kita bisa melakukan simulasi HTTP Request menggunakan NodeJS
- Terdapat 2 jenis module HTTP di NodeJS, HTTP dan HTTPS
- <https://nodejs.org/docs/latest-v22.x/api/http.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/http.html>
- <https://nodejs.org/docs/latest-v22.x/api/https.html>
- <https://nodejs.org/dist/latest-v16.x/docs/api/https.html>

Kode : HTTP Client

```
http-client.mjs
1 import https from "https";
2
3 const url = "https://hookb.in/1gmqywqrLLfd6N0061k8";
4 const request = https.request(url, options: {
5   method: "POST",
6   headers: {
7     "Content-Type": "application/json",
8     "Accept": "application/json",
9   }
10 }, callback: function (response : IncomingMessage ) {
11   response.addListener( event: "data", listener: function (data) {
12     console.info( data: `Receive : ${data.toString()}` )
13   })
14 });
```

trigger x

CONFIGURE

SELECT EVENT

The unique URL to trigger this workflow is:

<https://eodxx6cjlw7u0bw2.m.pipedream.net>

[Generate Test Event](#) [See Code Examples](#)

Action Required

Send an HTTP request to the URL above or generate a test event. [More](#)

☐ Waiting for events...

The selected event provides data for autocomplete and testing to help you build your workflow

```
16 const body = JSON.stringify( value: {
17   firstName: "Eko",
18   lastName: "Khannedy",
19 });
20
21 request.write(body);
22 request.end();
23
```

RequestBin

v1

Active



LIVE EVENTS



Hari Ini



POST /

06.19.07



POST /

06.18.35



30YNB00D8XEX0IDGVVYMLRR5ZS2

Success

Workflow executed in 745 ms

[▶ details](#)

trigger

Exports

Inputs

Logs

Details

▼ steps.trigger {2}

▸ context {19}

▼ event {7}

method: POST

path: /

▼ query {0} [Copy Path](#) · [Copy Value](#)

client_ip: 182.8.226.34

url: https://eolotbrqd37f7rl.m.pipedream.net/

▸ headers {4}

▼ body {2}

firstName: Asroni

lastName: Sukirman

HTTP Server



HTTP Server

- Standard Library HTTP juga tidak hanya bisa digunakan untuk membuat HTTP Client, tapi juga bisa digunakan untuk membuat HTTP Server
- Untuk kasus sederhana, cocok sekali jika ingin membuat HTTP Server menggunakan standard library NodeJS, namun untuk kasus yang lebih kompleks, direkomendasikan menggunakan library atau framework yang lebih mudah penggunaannya
- <https://nodejs.org/dist/latest-v16.x/docs/api/http.html>



Kode : Simple HTTP Server

```
http-server.mjs x
1 import http from "http";
2
3 const server = http.createServer( requestListener: (request : IncomingMessage , response : ServerResponse ) => {
4     response.write( chunk: "Hello World");
5     response.end();
6 });
7
8 server.listen( port: 3000);
9
```

Kode : Request Response HTTP Server

```
1 import http from "http";
2
3 const server = http.createServer( requestListener: (request : IncomingMessage , response : ServerResponse ) => {
4   if (request.method === "POST") {
5     request.addListener( event: "data", listener: function (data) {
6       response.setHeader( name: "Content-Type", value: "application/json");
7       response.write(data);
8       response.end();
9     })
10   } else {
11     response.write( chunk: "Hello World");
12     response.end();
13   }
14 });
15
16 server.listen( port: 3000);
17
```




http://localhost:3000/roni

Save

GET



http://localhost:3000/roni

Send



Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body

Cookies

Headers (4)

Test Results



Status: 200 OK

Time: 22 ms

Size: 143 B

Save Response



Pretty

Raw

Preview

Visualize

Text



1 Hello Asroni



http://localhost:3000/eko

Save

GET



http://localhost:3000/eko

Send



Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body

Cookies

Headers (4)

Test Results



Status: 200 OK

Time: 3 ms

Size: 148 B

Save Response



Pretty

Raw

Preview

Visualize

Text



1 Hello HTTP Server

Cluster



Cluster

- Seperti yang dijelaskan di awal, bahwa NodeJS itu secara default dia berjalan single thread, kecuali jika kita membuat thread manual menggunakan worker thread, tapi tetap dalam satu process
- NodeJS memiliki standard library bernama Cluster, dimana kita bisa menjalankan beberapa process NodeJS secara sekaligus
- Ini sangat cocok ketika kita menggunakan CPU yang multicore, sehingga semua core bisa kita utilisasi dengan baik, misal kita jalankan process NodeJS sejumlah CPU core
- <https://nodejs.org/dist/latest-v16.x/docs/api/cluster.html>



Cluster Primary dan Worker

- Di dalam Cluster, terdapat 2 jenis aplikasi, Primary dan Worker
- Primary biasanya digunakan sebagai koordinator atau manajer untuk para Worker
- Sedangkan Worker sendiri adalah aplikasi yang menjalankan tugas nya

Kode : Cluster Primary

```
cluster.mjs x
1 import cluster from "cluster";
2 import http from "http";
3 import os from "os";
4 import process from "process";
5
6 if (cluster.isPrimary) {
7   for (let i = 0; i < os.cpus().length; i++) {
8     cluster.fork();
9   }
10  cluster.addListener( event: "exit", listener: function (worker : Worker ) {
11    console.info( data: `Worker ${worker.id} is exited`);
12  })
13 }
14
```

Kode : Cluster Worker

```
14
15 if (cluster.isWorker) {
16   const server = http.createServer( requestListener: (request : IncomingMessage , response : ServerResponse ) => {
17     response.write( chunk: `Response from process ${process.pid}`);
18     response.end();
19     process.exit();
20   })
21   server.listen( port: 3000);
22   console.info( data: `Start cluster worker ${process.pid}`);
23 }
```

← → ↺ http://localhost:3000
Response from process 25902

```
worker : 25902
Worker-12 is exit
worker : 25905
Worker-13 is exit
worker : 25909
Worker-14 is exit
worker : 25910
Worker-15 is exit
worker : 25911
Worker-16 is exit
worker : 25914
Worker-17 is exit
worker : 25918
Worker-18 is exit
worker : 25919
Worker-19 is exit
worker : 25920
```

Materi Selanjutnya



Materi Selanjutnya

- NPM (Node Package Manager)
- NodeJS Unit Test
- ExpressJS
- NodeJS Database
- Dan lain-lain