

Sep. 21/09 | 2023

NodeJS Logging

Eko Kurniawan Khannendy

Eko Kurniawan Khannedy

- Technical architect at one of the biggest ecommerce company in Indonesia
- 11+ years experiences
- www.programmerzamannow.com
- youtube.com/c/ProgrammerZamanNow



Eko Kurniawan Khannedy

- Telegram : [@khannedy](https://t.me/khannedy)
- Facebook : [fb.com/ProgrammerZamanNow](https://www.facebook.com/ProgrammerZamanNow)
- Instagram : [instagram.com/programmerzamannow](https://www.instagram.com/programmerzamannow)
- Youtube : [youtube.com/c/ProgrammerZamanNow](https://www.youtube.com/c/ProgrammerZamanNow)
- Telegram Channel : t.me/ProgrammerZamanNow
- Email : echo.khannedy@gmail.com

Sebelum Belajar

- Sudah Mengikuti Roadmap Kelas JavaScript dari Programmer Zaman Now ✓
- NodeJS Dasar ✓
- NodeJS Unit Test ✓

Agenda

- Pengenalan Logging
- Logging Library
- Logger
- Transport
- Format
- Dan lain-lain

Junix 21/15/2023

WNB

- vt -

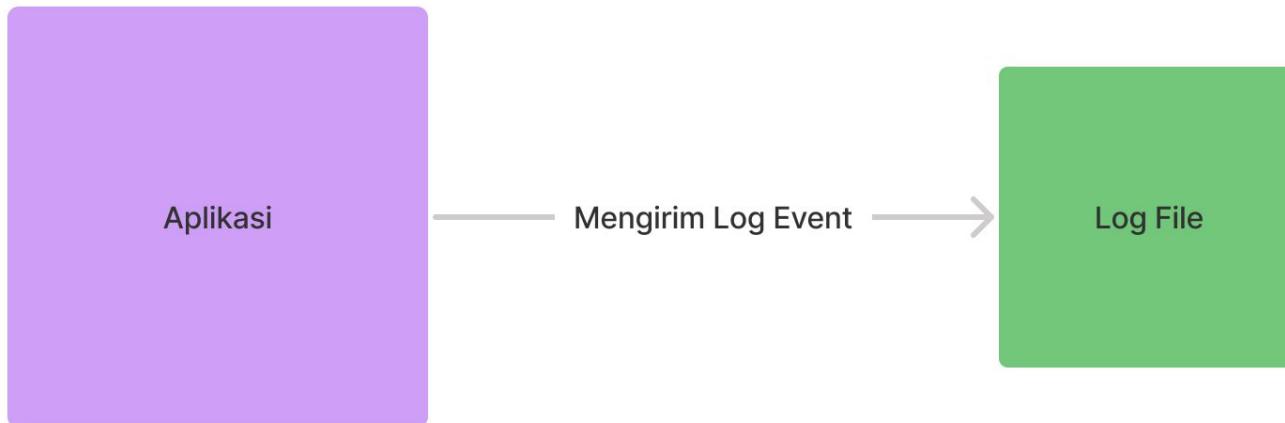
ELEMOSBY

Pengenalan Logging

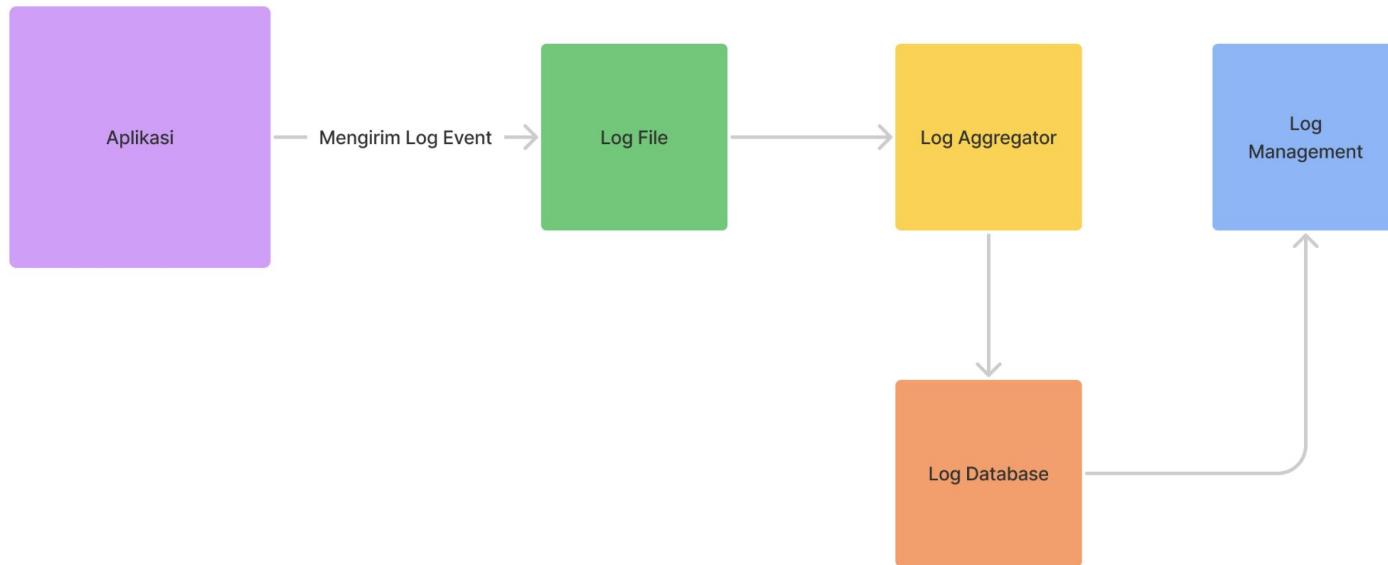
Pengenalan Logging

- Log file adalah file yang berisikan informasi kejadian dari sebuah sistem
- Biasanya dalam log file, terdapat informasi waktu kejadian dan pesan kejadian
- Logging adalah aksi menambah informasi log ke log file
- Logging sudah menjadi standard industri untuk menampilkan informasi yang terjadi di aplikasi yang kita buat
- Logging bukan hanya untuk menampilkan informasi, kadang digunakan untuk proses debugging ketika terjadi masalah di aplikasi kita

Diagram Logging



Ekosistem Logging



Logging Library

NodeJS Logging

- NodeJS sendiri sebenarnya memilih fitur untuk melakukan logging dengan object console
- Namun saat ini, kebanyakan programmer hanya menggunakan fitur ini pada kasus-kasus yang sederhana
- Hal ini dikarenakan penggunaannya yang kurang flexible dan minim fitur
- <https://nodejs.org/api/console.html>

Logging Library

Diluar NodeJS Logging, banyak sekali library yang bisa kita gunakan untuk logging, seperti :

- Winston <https://www.npmjs.com/package/winston>
- Bunyan <https://www.npmjs.com/package/bunyan>
- Pino <https://www.npmjs.com/package/pino>
- Loglevel <https://www.npmjs.com/package/loglevel>
- NPMLog <https://www.npmjs.com/package/npmlog>

Winston

- Pada kelas ini kita akan menggunakan Winston
- Winston merupakan logging library yang sangat populer di kalangan programmer NodeJS
- <https://www.npmjs.com/package/winston>
- <https://github.com/winstonjs/winston>

30 mis 21/09/2017

Wibowo

- VS -

EZYBOMY

Membuat Project

Membuat Project

- <https://github.com/ProgrammerZamanNow/belajar-nodejs-unit-test>

Menambah Dependency Winston

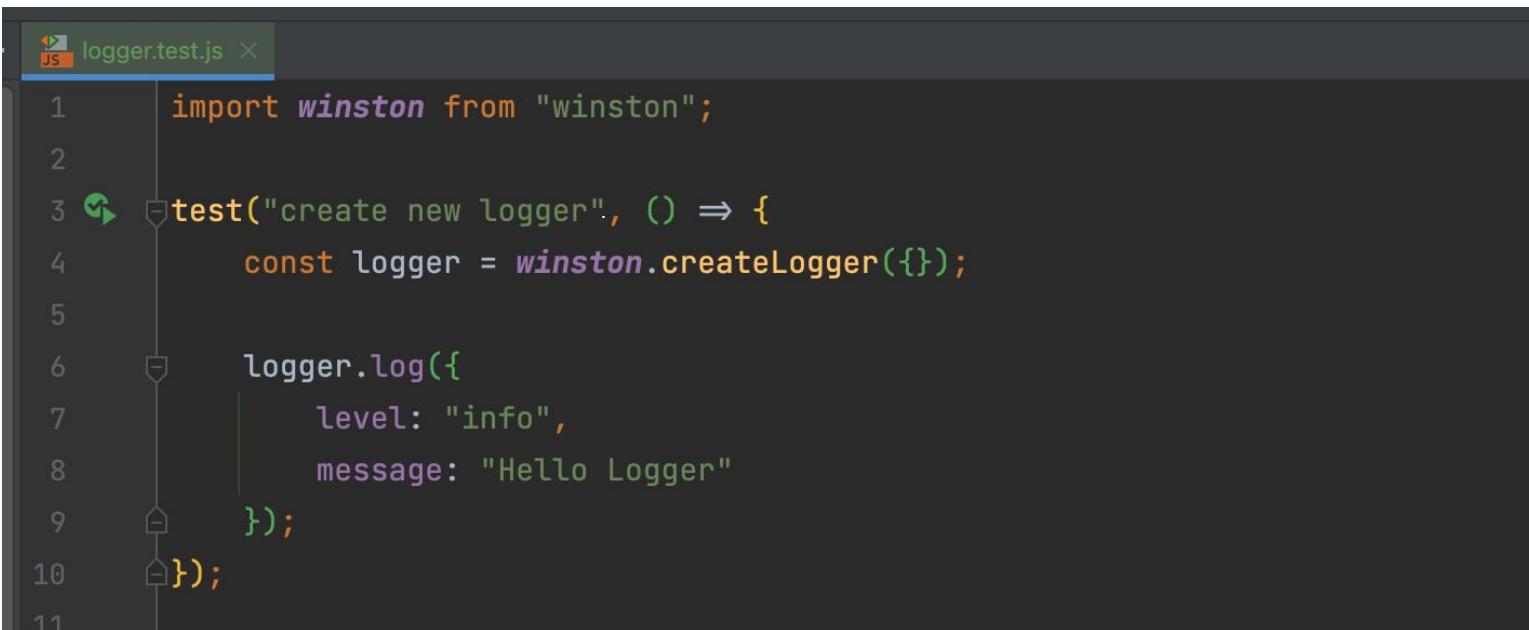
```
1  "license": "ISC",
2
3  "dependencies": {
4      "winston": "^3.7.2"
5  },
6
7  "devDependencies": {
8      "@babel/plugin-transform-runtime": "^7.17.12",
9      "@babel/preset-env": "^7.17.12",
10     "babel-jest": "^28.1.0",
11     "jest": "^28.1.0"
12 }
```

Logger

Logger

- Logger adalah sebuah object di Winston, yang digunakan untuk melakukan logging
- Untuk membuat object Logger, kita bisa menggunakan function `createLogger()` yang terdapat di package/module `winston`

Kode : Membuat Logger



The screenshot shows a code editor window with a dark theme. The file tab at the top is labeled "logger.test.js". The code itself is a Jest test for the Winston logger library. It imports Winston and creates a new logger with an info level and a message. The code is numbered from 1 to 11.

```
1 import winston from "winston";
2
3 test("create new logger", () => {
4     const logger = winston.createLogger({});
5
6     logger.log({
7         level: "info",
8         message: "Hello Logger"
9     });
10});
11
```

Console Transport

Console Transport

- Saat membuat Logger, secara default Logger tidak akan menggunakan Transport
- Apa itu Transport? Transport adalah destinasi atau target yang digunakan untuk mengirim log.
- Ada banyak sekali Transport, salah satunya yang paling sederhana adalah **Console**
- Console merupakan Transport yang digunakan untuk mengirim data log ke console/stdout



Kode : Console Transport

```
2
3  ↗ test("create new logger with console transport", () => {
4      ↗ const logger = winston.createLogger({
5          ↗ transports: [
6              ↗     new winston.transports.Console({})
7          ↗ ]
8      });
9
10     ↗ logger.log({
11         ↗     level: "info",
12         ↗     message: "Hello Logger"
13     });
14 });
15 }
```

Level

- Dalam logging, terdapat istilah yang bernama Level, Level biasanya digunakan sebagai informasi seberapa penting log tersebut
- Level dimulai dari paling rendah ke paling tinggi, semakin tinggi Level nya, artinya semakin penting informasi log tersebut
- Untuk menambahkan Level ketika melakukan log, kita bisa ubah attribute level menjadi level yang kita inginkan

Logging Level

Winston Level

error	/
warn	/
info	/
http	/
verbose	/
debug	/
silly	/

Handwritten annotations in orange ink:

- A vertical line is drawn through the center of the table, passing through all seven rows.
- The word "Plenty" is written vertically next to the line, centered between the "info" and "http" rows.



Kode : Level

```
3  ↗  ⌂ test("logging with level", () => {
4    const logger = winston.createLogger({
5      transports: [
6        new winston.transports.Console({})
7      ]
8    });
9
10   logger.log({level: "error", message: "Error Message"});
11   logger.log({level: "warn", message: "Warn Message"});
12   logger.log({level: "info", message: "Info Message"});
13   logger.log({level: "http", message: "HTTP Message"});
14   logger.log({level: "verbose", message: "Verbose Message"});
15   logger.log({level: "debug", message: "Debug Message"});
16   logger.log({level: "silly", message: "Silly Message"});
17 });


```



Kode : Hasil Log Level

```
✓ Tests passed: 1 of 1 test – 4 ms
ms  jest --testNamePattern=logging with level
ms
  {"level":"error","message":"Error Message"}
  {"level":"warn","message":"Warn Message"}
  {"level":"info","message":"Info Message"}
```

Kenapa Beberapa Level Tidak Muncul?

- Secara default, saat kita membuat Logger, Logger hanya akan menampilkan log dengan level info atau level diatasnya
- Jika kita ingin mengubah log level mana yang ingin kita tampilkan, maka kita perlu ubah konfigurasi level ketika kita membuat logger, dengan menggunakan level minimal yang ingin kita tampilkan

Kode : Logger dengan Level

```
3  ↵  test("logging with level", () => {
4    const logger = winston.createLogger({
5      level: "debug",
6      transports: [
7        new winston.transports.Console({})
8      ]
9    });
0
1    logger.log({level: "error", message: "Error Message"});
2    logger.log({level: "warn", message: "Warn Message"});
3    logger.log({level: "info", message: "Info Message"});
4    logger.log({level: "http", message: "HTTP Message"});
5    logger.log({level: "verbose", message: "Verbose Message"});
6    logger.log({level: "debug", message: "Debug Message"});
7    logger.log({level: "silly", message: "Silly Message"});
8  });
9
```

Shortcut Function

- Logger juga memiliki function shortcut yang bisa digunakan untuk melakukan logging, sehingga kita tidak perlu menggunakan function log dan object dengan attribute level lagi
- Kita bisa langsung menggunakan function dengan nama sesuai dengan level nya, misal logger.info(message), logger.warn(message), dan lain-lain

Kode : Shortcut Function

```
↻ test("logging with shortcut level function", () => {
    const logger = winston.createLogger({
        level: "silly",
        transports: [
            new winston.transports.Console({})
        ]
    });

    logger.error("Error Message");
    logger.warn("Warn Message");
    logger.info("Info Message");
    logger.http("HTTP Message");
    logger.verbose("Verbose Message");
    logger.debug("Debug Message");
    logger.silly("Silly Message");
});
```

Format



Format

- Format adalah object yang digunakan untuk melakukan formatting data log sebelum dikirim ke Transport
- Saat kita membuat Logger, secara default, Logger akan menggunakan JSON Format, artinya data akan dikirim dalam bentuk JSON
- Winston juga menyediakan banyak format lain selain JSON



Kode : Format

```
39  ↵  ↴ test("logging with format", () => {
40    ↵    ↴ const logger = winston.createLogger({
41      ↵      ↴   level: "info",
42      ↵      ↴   // format: winston.format.json(),
43      ↵      ↴   format: winston.format.simple(),
44      ↵      ↴   transports: [
45        ↵        ↴     new winston.transports.Console({})
46        ↵        ↴   ]
47      ↵      ↴ });
48
49      ↵      ↴ logger.info("Hello World");
50  ↵  ↴});
```

Membuat Format Sendiri

- Winston juga menyediakan format printf, yang bisa digunakan untuk membuat format sendiri

Kode : Printf Format

```
38
39 ↗ 39 test("logging with printf format", () => {
40   ↗ 40     const logger = winston.createLogger({
41       ↗ 41       level: "info",
42       ↗ 42       format: winston.format.printf(info => {
43           ↗ 43           return `${new Date()} : ${info.level.toUpperCase()} : ${info.message}`;
44       },
45       ↗ 45       transports: [
46           ↗ 46           new winston.transports.Console({})
47       ],
48   });
49
50   ↗ 50   logger.info("Hello World");
51 });
52
```

Sep. 21 / 09 / 2023

Combine Format

Combine Format

- Winston memiliki fitur bernama Combine Format, dimana kita bisa melakukan kombinasi beberapa Format sekaligus
- Ini cocok untuk menambah informasi tambahan ke log data, misal data timestamp, data jarak waktu antar log, dan lain-lain
- Kita bisa menggunakan Combine Format di Winston untuk menggabungkan beberapa Format



Daftar Format di Winston

```
export namespace format {
    function align(): Format;
    function cli(opts?: CliOptions): Format;
    function colorize(opts?: ColorizeOptions): Colorizer;
    function combine(...formats: Format[]): Format;
    function errors(opts?: object): Format;
    function json(opts?: JsonOptions): Format;
    function label(opts?: LabelOptions): Format;
    function logstash(): Format;
    function metadata(opts?: MetadataOptions): Format;
    function ms(): Format;
    function padLevels(opts?: PadLevelsOptions): Format;
    function prettyPrint(opts?: PrettyPrintOptions): Format;
    function printf(templateFunction: (info: TransformableInfo) => string): Format;
    function simple(): Format;
    function splat(): Format;
    function timestamp(opts?: TimestampOptions): Format;
    function uncolorize(opts?: UncolorizeOptions): Format;
}
```

Kode : Combine Format

```
9  ⏎  test("logging with combine format", () => {
10  ⏎    const logger = winston.createLogger({
11  ⏎      level: "info",
12  ⏎      format: winston.format.combine(
13  ⏎        [
14  ⏎          winston.format.timestamp(),
15  ⏎          winston.format.ms(),
16  ⏎          winston.format.json()
17  ⏎        ],
18  ⏎        transports: [
19  ⏎          new winston.transports.Console({})
20  ⏎        ]
21  ⏎    });
22  ⏎  });
23  ⏎
```

File Transport

File Transport

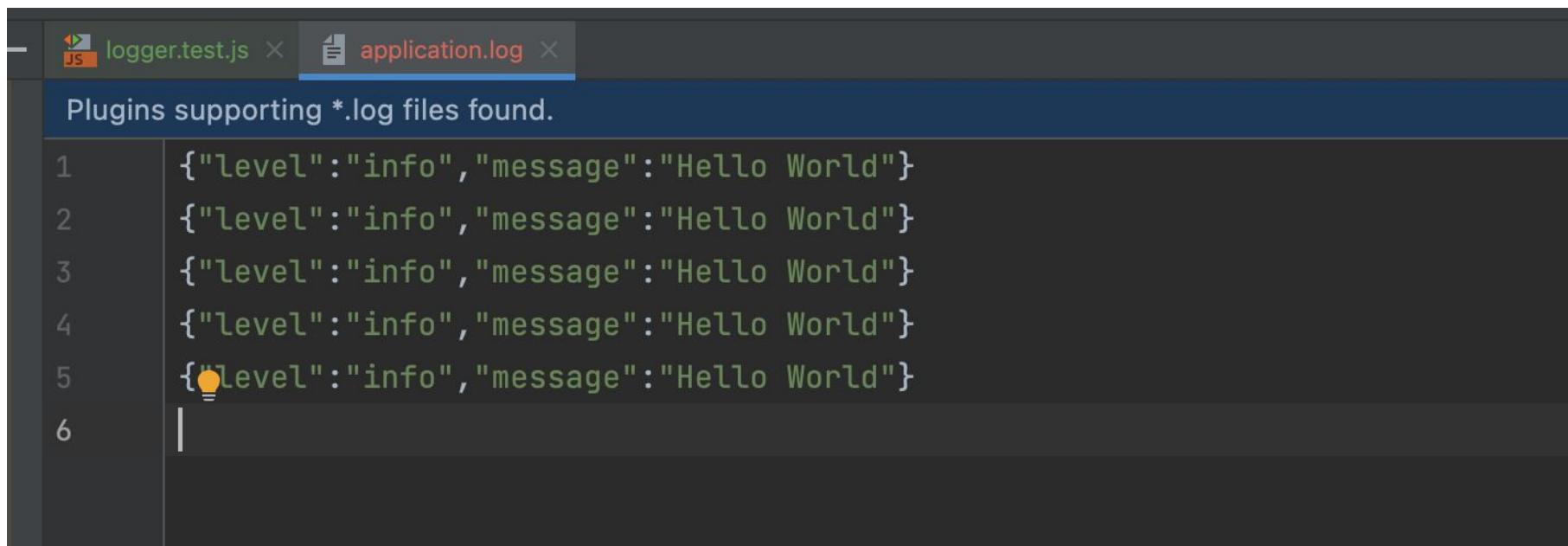
- Sebelumnya kita hanya menggunakan Console Transport, selain Console Transport, di Winston juga terdapat File Transport
- Dari namanya kita sudah tahu, bahwa Transport ini akan menyimpan data log ke file
- Kita bisa menambahkan langsung beberapa Transport dalam satu Logger object

Kode : File Transport

```
9  ↵  test("logging with file transport", () => {
0    ↵    const logger = winston.createLogger({
1      ↵      level: "info",
2      ↵      transports: [
3        ↵        new winston.transports.Console({}),
4        ↵        new winston.transports.File({
5          ↵            filename: "application.log"
6        ↵      },
7      ↵    ],
8    ↵  });
9
0    ↵    logger.info("Hello World");
1    ↵    logger.info("Hello World");

```

Hasil File Transport



A screenshot of a terminal window showing the contents of a file named "application.log". The window has two tabs at the top: "logger.test.js" and "application.log". The "application.log" tab is active, indicated by a blue underline. The text in the terminal shows five lines of JSON log entries, each with a line number on the left and the log message on the right. The fifth line contains a yellow lightbulb icon, indicating a potential issue or suggestion.

```
1 {"level": "info", "message": "Hello World"}  
2 {"level": "info", "message": "Hello World"}  
3 {"level": "info", "message": "Hello World"}  
4 {"level": "info", "message": "Hello World"}  
5 {"level": "info", "message": "Hello World"}  
6 |
```

becoming
2/04 | 2023

Transport Level

Transport Level

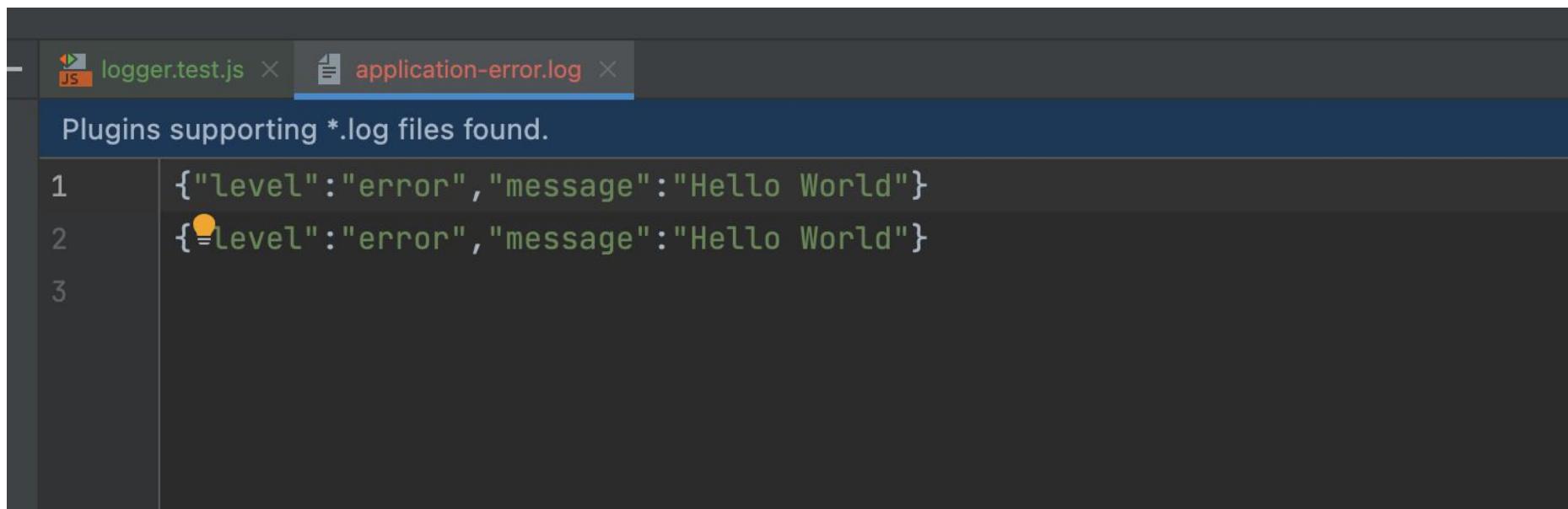
- Beberapa Transport memiliki pengaturan Level sendiri
- Saat sebuah Transport memiliki pengaturan Level, secara otomatis Transport hanya akan menerima log dengan level tersebut atau yang lebih tinggi
- Hal ini sangat cocok misal ketika kita ingin memisahkan beberapa level log di transport yang berbeda



Kode : Transport Level

```
▶ test("logging with file transport level", () => {
    const logger = winston.createLogger({
        level: "info",
        transports: [
            new winston.transports.Console({}),
            new winston.transports.File({
                filename: "application.log"
            }),
            new winston.transports.File({
                level: "error",
                filename: "application-error.log"
            }),
        ],
    });
});
```

Hasil File Transport

A screenshot of a code editor interface. At the top, there are two tabs: "logger.test.js" and "application-error.log". The "application-error.log" tab is active, indicated by a blue bar underneath it. Below the tabs, a message says "Plugins supporting *.log files found." followed by three numbered log entries. The first entry is a valid JSON object. The second entry has a syntax error where the double quotes around "Level" are swapped. The third entry is blank.

```
1 {"level": "error", "message": "Hello World"}  
2 {"Level": "error", "message": "Hello World"}  
3
```

Kemps 21/4/2023
Mon 08:00
- VS -
ELETRABRAS

Rotate File

Default Transport File

- Secara default, winston Transport File hanya akan menyimpan semua log di dalam satu file
- Hal ini akan sangat berbahaya ketika aplikasi berjalan dalam jangka waktu lama, sehingga menyebabkan ukuran file akan semakin **membesar**

Daily Rotate File

- Untungnya, Winston sendiri membuat package terpisah untuk membantu ini, yaitu Daily Rotate File
- <https://www.npmjs.com/package/winston-daily-rotate-file>
- Package ini bisa digunakan untuk membuat rotasi file transport secara otomatis sesuai aturan yang kita tentukan, dan bisa secara otomatis menghapus file lama ketika sudah tidak diperlukan lagi

Kode : Menambah Dependency

```
31      "license": "ISC",
32      "dependencies": {
33        "winston": "^3.7.2",
34        "winston-daily-rotate-file": "^4.7.1"
35      },
36      "devDependencies": {
37        "@babel/plugin-transform-runtime": "^7.17.12",
38        "@babel/preset-env": "^7.17.12",
39        "babel-jest": "^28.1.0",
40        "jest": "^28.1.0"
```

Kode : Rotate File

```
2 import DailyRotateFile from 'winston-daily-rotate-file';
3
4 test("logging with daily rotate file", () => {
5     const logger = winston.createLogger({
6         level: "info",
7         transports: [
8             new winston.transports.Console({}),
9             new DailyRotateFile({
10                 filename: 'app-%DATE%.log',
11                 zippedArchive: true,
12                 maxSize: '1m',
13                 maxFiles: '14d'
14             })
15         ]
16     });
17
18     for (let i = 0; i < 100000; i++) {
19         logger.info("Hello World");
20     }
}
```

} Daily Rotate file.

Membuat Transport

Membuat Transport

- Kadang pada kasus tertentu, mungkin kita ingin membuat Transport sendiri, misal ketika ada log error, kita ingin mengirim data log tersebut ke database, atau ke chat, atau email, dan lain-lain
- Jika tidak ada Transport yang tersedia secara default oleh Winston, kita bisa mencari Transport yang opensource yang disediakan di komunitas, atau bisa saja kita membuat Transport sendiri secara manual jika kita mau
- Untuk membuat Transport, kita cukup membuat class turunan dari package `winston-transport`

Spesifikasi Transport

```
declare class TransportStream extends stream.Writable {  
  public format?: logform.Format;  
  public level?: string;  
  public silent?: boolean;  
  public handleExceptions?: boolean;  
  public handleRejections?: boolean;  
  
  constructor(opts?: TransportStream.TransportStreamOptions);  
  
  public log?(info: any, next: () => void): any;  
  public logv?(info: any, next: () => void): any;  
  public close?(): void;  
}
```

```
interface TransportStreamOptions {  
  format?: logform.Format;  
  level?: string;  
  silent?: boolean;  
  handleExceptions?: boolean;  
  handleRejections?: boolean;
```

Kode : Membuat Transport

```
import DailyRotateFile from 'winston-daily-rotate-file';
import TransportStream from "winston-transport"

test("create new transport", () => {

    class MyTransport extends TransportStream {

        constructor(option) {
            super(option);
        }

        log(info, next) {
            console.log(` ${new Date()} : ${info.level.toUpperCase()} : ${info.message}`);
            next();
        }
    }
})
```



Kode : Menggunakan Transport

```
const logger = winston.createLogger({
  level: "silly",
  transports: [
    new MyTransport({})
  ]
});

logger.error("Error Message");
logger.warn("Warn Message");
logger.info("Info Message");
logger.http("HTTP Message");
logger.verbose("Verbose Message");
logger.debug("Debug Message");
logger.silly("Silly Message");
```

Transport Lainnya

- Winston juga menyediakan Transport lain, misalnya :
 - Redis Transport : <https://github.com/winstonjs/winston-redis>
 - Syslog Transport : <https://github.com/winstonjs/winston-syslog>
 - CouchDB Transport : <https://github.com/winstonjs/winston-couchdb>
 - Loggy Transport : <https://github.com/winstonjs/winston-loggly>
- Atau Transport yang disediakan oleh komunitas :
 - Slack Transport : <https://github.com/TheAppleFreak/winston-slack-webhook-transport>
 - Telegram Transport : <https://github.com/ivanmarban/winston-telegram>

Exceptions

Exceptions

- Saat kita membuat program NodeJS, kadang kita lupa menangkap Exception
- Hal ini bisa berbahaya karena nanti kita tidak bisa melakukan debug Exception dengan baik, sehingga tidak bisa kita investigasi selanjutnya

Kode : Exception

The screenshot shows a code editor interface with a dark theme. At the top, there is a status bar with a green progress bar. Below it, the title bar displays "logging-exception.js" with a JS icon. The main workspace contains two lines of code:

```
1  hello();  
2
```

A tooltip "You, Today • Uncommitted changes" is visible near the bottom of the code area. The bottom half of the screen shows the terminal output:

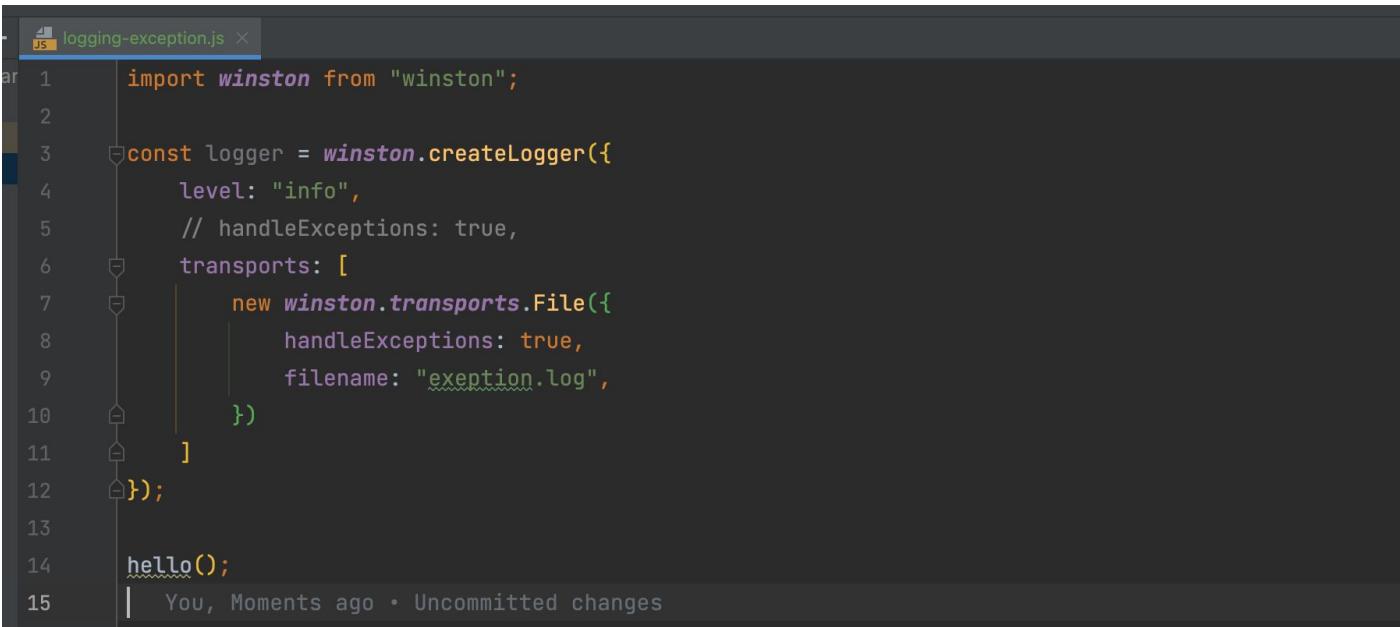
```
/Users/khannedy/.nvm/versions/node/v18.15.0/bin/node /Users/khannedy/Developments/BELAJAR/NODEJS/belajar-nodejs-logging  
file:///Users/khannedy/Developments/BELAJAR/NODEJS/belajar-nodejs-logging/src/logging-exception.js:1  
hello();  
^  
ReferenceError: hello is not defined  
at file:///Users/khannedy/Developments/BELAJAR/NODEJS/belajar-nodejs-logging/src/logging-exception.js:1:1  
at ModuleJob.run (node:internal/modules/esm/module_job:185:25)  
at async Promise.all (index 0)
```

Handle Exception

- Winston memiliki fitur secara otomatis yang bisa digunakan untuk menangkap Exception yang belum ter-handle
- Kita bisa lakukan pengaturan ini di Logger, yang secara otomatis exception akan dikirim ke semua Transport
- Atau kita bisa lakukan pengaturan ini di Transport



Kode : Handle Exceptions



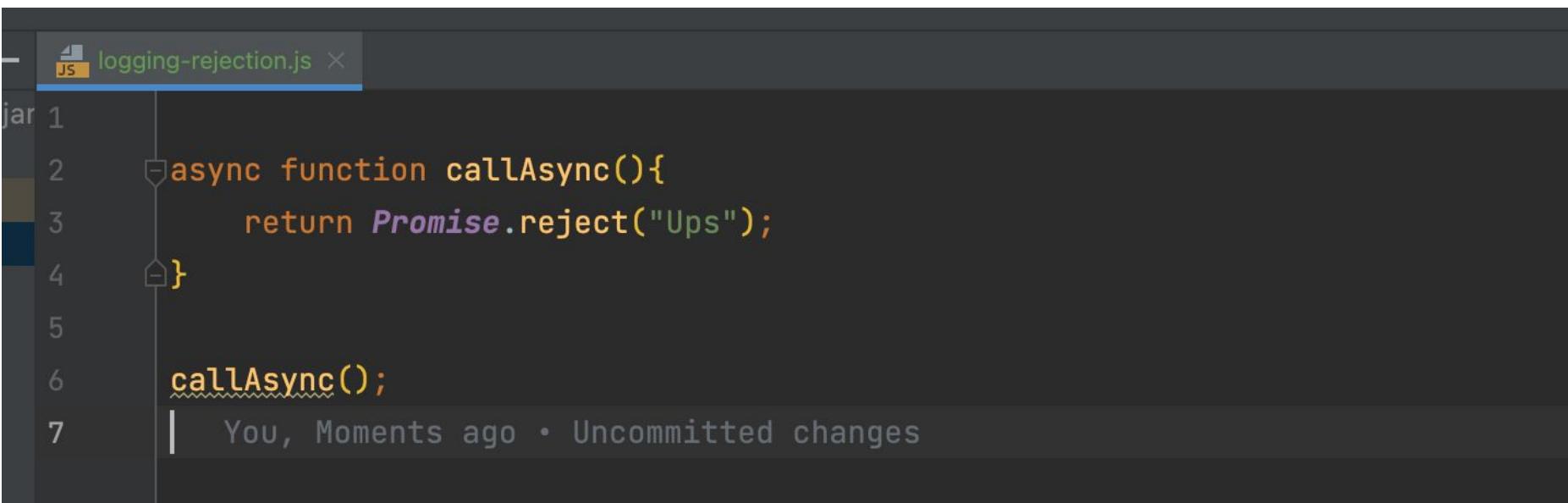
```
logging-exception.js ×
1 import winston from "winston";
2
3 const logger = winston.createLogger({
4   level: "info",
5   // handleExceptions: true,
6   transports: [
7     new winston.transports.File({
8       handleExceptions: true,
9       filename: "exception.log",
10      })
11    ],
12  });
13
14 hello();
15 | You, Moments ago • Uncommitted changes
```

Rejections

Rejections

- Selain kasus Exception yang tidak terhandle, kadang sering ada kasus di NodeJS kita sering lupa meng-handle Promise Rejection

Kode : Rejections



```
logging-rejection.js ×
jar 1
2     async function callAsync(){
3         return Promise.reject("Ups");
4     }
5
6     callAsync();
7 | You, Moments ago • Uncommitted changes
```

The screenshot shows a code editor window with a dark theme. The file tab at the top is labeled "logging-rejection.js". The code itself is as follows:

```
async function callAsync(){
    return Promise.reject("Ups");
}

callAsync();
```

A cursor is positioned at the end of the second line of code. A status bar at the bottom right indicates "You, Moments ago • Uncommitted changes".

Handle Promise Rejections

- Winston juga memiliki fitur yang bisa kita gunakan untuk menangkap semua Promise yang reject secara otomatis
- Jadi kita bisa melihat detail error Rejections tersebut
- Sama seperti Exceptions, kita bisa atur ini di Logger atau di Transport



Kode : Handle Promise Rejections

```
const logger = winston.createLogger({
  level: "info",
  // handleExceptions: true,
  // handleRejections: true,
  transports: [
    new winston.transports.File({
      handleExceptions: true,
      handleRejections: true,
      filename: "exception.log",
    })
  ],
});

callAsync();
```

You, Moments ago • Uncommitted changes

Materi Selanjutnya

Materi Selanjutnya

- Express JS
- NodeJS Database
- Dan lain-lain