PLMS Assignment 5

PLMS - Building Microservices using NodeJS Assignments

Task: Implement logging into your services.

Task. Implement logging into your services

In this task you will be responsible for logging the data. Previously you may have used `console.log` to check where you have an error, which is okay in development environment but not in deployment environment, we simply cannot use `console.log` to make our logger system work.

Step 1. Set up Winston

Install “winston" using npm. We will be using this to log the data properly.

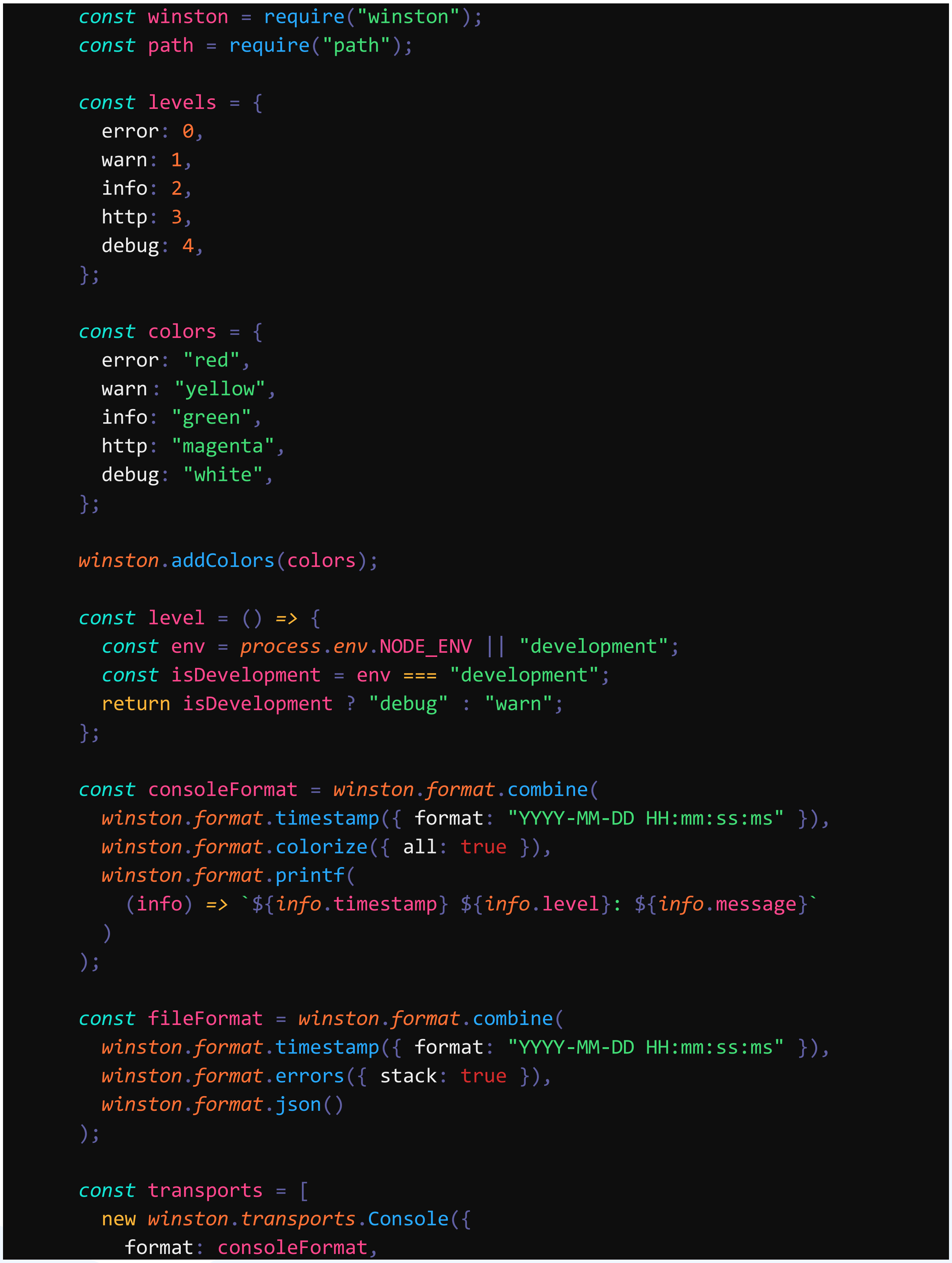
```
npm install winston
```

Step 2. We will now work on a utility for our logger.

Create a new file under `src/untils`, called `logger.js`.

In this file, we will be making guidelines for our logger. These guidelines includes level, colors and other standards for logging.

Here is the logger file that you can use:



```
const winston = require("winston");
const path = require("path");

const levels = {
  error: 0,
  warn: 1,
  info: 2,
  http: 3,
  debug: 4,
};

const colors = {
  error: "red",
  warn : "yellow",
  info: "green",
  http: "magenta",
  debug: "white",
};

winston.addColors(colors);

const level = () => {
  const env = process.env.NODE_ENV || "development";
  const isDevelopment = env === "development";
  return isDevelopment ? "debug" : "warn";
};

const consoleFormat = winston.format.combine(
  winston.format.timestamp({ format: "YYYY-MM-DD HH:mm:ss:ms" }),
  winston.format.colorize({ all: true }),
  winston.format.printf(
    (info) => `${info.timestamp} ${info.level}: ${info.message}`
  )
);

const fileFormat = winston.format.combine(
  winston.format.timestamp({ format: "YYYY-MM-DD HH:mm:ss:ms" }),
  winston.format.errors({ stack: true }),
  winston.format.json()
);

const transports = [
  new winston.transports.Console({
    format: consoleFormat,
    level: level(),
  }),

  new winston.transports.File({
    filename: path.join(__dirname, "../../logs/error.log"),
    level: "error",
    format: fileFormat,
  }),

  new winston.transports.File({
    filename: path.join(__dirname, "../../logs/combined.log"),
    format: fileFormat,
  }),
];

const logger = winston.createLogger({
  level: level(),
  levels,
  format: fileFormat,
  transports,
});

module.exports = logger;
```

Use this file in all your services. After that you can start changing all your `console.log` into logger functions to start logging each action and events that happen in all your service.

Note: *You need to do this for all the services we have worked on so far.*

Step 3. Once all these functionalities are built successfully then you run your service again and you see a new folder called `log`. There would be 2 different files one for all longs and one for only the errors. This way you can pinpoint the errors more effectively.

If you can see the logs in your log file then congratulations, you have successfully created a logger for your service. You can now use this log file to properly demonstrate any errors you get into your services.

Congrats! You have successfully completed this 5-day intensive React training!

We really appreciate your support and hardworking throughout the week. Hope you have enjoyed React!

Have a good weekend and see you next time!