# Console

const { Console } = require('console');

const output = fs.createWriteStream('./stdout.log');

const errorOutput = fs.createWriteStream('./stderr.log');

const logger = new Console({ stdout: output, stderr: errorOutput });

logger.log('count: %d', 5);

// In stdout.log: count 5

console.debug()

console.info()

console.log('count: %d', count);

console.trace() // show cả printStackTrace

console.warn()

console.error(new Error('Whoops, something bad happened'));

// Prints: [Error: Whoops, something bad happened], to stderr

console.table([{ a: 1, b: 'Y' }, { a: 'Z', b: 2 }]);

// ┌─────────┬─────┬─────┐

// │ (index) │  a  │  b  │

// ├─────────┼─────┼─────┤

// │    0    │  1  │ 'Y' │

// │    1    │ 'Z' │  2  │

// └─────────┴─────┴─────┘

console.table([{ a: 1, b: 'Y' }, { a: 'Z', b: 2 }], ['a']);

// ┌─────────┬─────┐

// │ (index) │  a  │

// ├─────────┼─────┤

// │    0    │  1  │

// │    1    │ 'Z' │

// └─────────┴─────┘

console.time('100-elements');

for (let i = 0; i < 100; i++) {}

console.timeEnd('100-elements');

// prints 100-elements: 225.438ms

# DNS

Dùng để look up IP address của host

const dns = require('dns');

dns.lookup('example.org', (err, address, family) => {

  console.log('address: %j family: IPv%s', address, family);

});

// address: "93.184.216.34" family: IPv4

// lookupService(address, port, (err, hostname, service)

dns.lookupService('127.0.0.1', 22, (err, hostname, service) => {

  console.log(hostname, service);

  // Prints: localhost ssh

});

const { Resolver } = require('dns');

const resolver = new Resolver();

resolver.setServers(['4.4.4.4']);

// This request will use the server at 4.4.4.4, independent of global settings.

resolver.resolve4('example.org', (err, addresses) => {

  // ...

});

/\*

[

    { type: 'A', address: '127.0.0.1', ttl: 299 },

    { type: 'CNAME', value: 'example.com' },

    { type: 'MX', exchange: 'alt4.aspmx.l.example.com', priority: 50 },

    { type: 'NS', value: 'ns1.example.com' },

    { type: 'TXT', entries: [ 'v=spf1 include:\_spf.example.com ~all' ] },

    {

        type: 'SOA',

        nsname: 'ns1.example.com',

        hostmaster: 'admin.example.com',

        serial: 156696742,

        refresh: 900,

        retry: 900,

        expire: 1800,

        minttl: 60

    }

]

\*/

resolver.getServers()

resolver.resolve()

resolver.resolve4()

resolver.resolve6()

resolver.resolveAny()

# Error

Class: Error

* new Error(message)
* Error.captureStackTrace(targetObject[, constructorOpt])
* Error.stackTraceLimit
* error.code
* error.message
* error.stack

Class: SystemError

* error.address
* error.code
* error.dest
* error.errno
* error.info
* error.message
* error.path
* error.port
* error.syscall

# Events

Class: EventEmitter

|  |  |
| --- | --- |
| * EventEmitter.defaultMaxListeners * EventEmitter.errorMonitor * emitter.addListener(eventName, listener) * emitter.emit(eventName[, ...args]) * emitter.eventNames() * emitter.getMaxListeners() * emitter.listenerCount(eventName) * emitter.listeners(eventName) * emitter.off(eventName, listener) * emitter.on(eventName, listener) * emitter.once(eventName, listener) * emitter.removeAllListeners([eventName]) * emitter.removeListener(eventName, listener) |  |
| * emitter.setMaxListeners(n) | Default : 10 listeners / event |

const EventEmitter = require("events");

emitter.setMaxListeners(emitter.getMaxListeners() + 1);

emitter.once("event", () => {

  // do stuff

  emitter.setMaxListeners(Math.max(emitter.getMaxListeners() - 1, 0));

});

eventNames()

const myEE = new EventEmitter();

const sym = Symbol("symbol");

myEE.on("foo", () => {});

myEE.on("bar", () => {});

myEE.on(sym, () => {});

console.log(myEE.eventNames());

// Prints: [ 'foo', 'bar', Symbol(symbol) ]

## Async vs Sync

EventEmitter luôn là sync ops vì tránh race condition & logic err

Nhưng nếu mình muốn async mode :

* setImmediate()
* process.nextTick()

const myEmitter = new MyEmitter();

myEmitter.on("event", (a, b) => {

  setImmediate(() => {

    console.log("this happens asynchronously");

  });

});

myEmitter.emit("event", "a", "b");

## Code Template

const EventEmitter = require('events');

const eventEmitter = new EventEmitter();

/\* 1. Emit with input  \*/

eventEmitter.on("sayHello", (input) => {

    console.log(`hello ${input}`);

})

eventEmitter.emit("sayHello", "trung thinh");

/\* 2. Emit with Class \*/

class Student extends EventEmitter {

    constructor(name) {

        super(); // dùng để inherit EventEmitter

        this.\_name = name;

    }

    get name() {

        return this.\_name;

    }

}

let newStudent = new Student("jungtin");

newStudent.on("name", function() {

    console.log(newStudent.name);

})

newStudent.emit("name");

# File System

Class: fs.Dir

dir.close(callback)

dir.path

dir.read(callback)

fs.mkdir(path[, options], callback)

fs.mkdtemp(prefix[, options], callback)

fs.open(path[, flags[, mode]], callback)

fs.opendir(path[, options], callback)

fs.readdir(path[, options], callback)

|  |  |
| --- | --- |
| Class: fs.ReadStream   * Event: 'close' * Event: 'open' * Event: 'ready' * readStream.bytesRead * readStream.path * readStream.pending | **Class: fs.WriteStream**   * Event: 'close' * Event: 'open' * Event: 'ready' * writeStream.bytesWritten * writeStream.path * writeStream.pending |

fs.createReadStream(path[, options])

fs.createWriteStream(path[, options])

* fs.write(fd, buffer[, offset[, length[, position]]], callback)
* fs.writeFile(file, data[, options], callback)
* fs.appendFile(path, data[, options], callback)
* fs.close(fd, callback)
* fs.copyFile(src, dest[, mode], callback)
* fs.exist(path)
  + Deprecated: Use [fs.stat()](https://nodejs.org/docs/latest-v15.x/api/fs.html#fs_fs_stat_path_options_callback) or [fs.access()](https://nodejs.org/docs/latest-v15.x/api/fs.html#fs_fs_access_path_mode_callback) instead.
* fs.link(existingPath, newPath, callback)
* fs.readFile(path[, options], callback)
* fs.read(fd, buffer, offset, length, position, callback)
* fs.readlink(path[, options], callback)

Descriptor

* fs.realpath(path[, options], callback)
* fs.realpath.native(path[, options], callback)
* fs.rename(oldPath, newPath, callback)
* fs.rmdir(path[, options], callback)
* fs.rm(path[, options], callback)
* fs.stat(path[, options], callback)
* fs.symlink(target, path[, type], callback)
* fs.truncate(path[, len], callback)
* fs.unlink(path, callback)

fs.open() nhớ close()

const fs = require('fs');

fs.open('/open/some/file.txt', 'r', (err, fd) => {

  if (err) throw err;

  fs.close(fd, (err) => {

    if (err) throw err;

  });

});

Copy File

const fs = require('fs');

const { COPYFILE\_EXCL } = fs.constants;

function callback(err) {

  if (err) throw err;

  console.log('source.txt was copied to destination.txt');

}

// destination.txt will be created or overwritten by default.

fs.copyFile('source.txt', 'destination.txt', callback);

// By using COPYFILE\_EXCL, the operation will fail if destination.txt exists.

fs.copyFile('source.txt', 'destination.txt', COPYFILE\_EXCL, callback);

* fs.constants.COPYFILE\_EXCL: sẽ không thực hiện nếu dest đã existed.
* fs.constants.COPYFILE\_FICLONE: The copy operation will attempt to create a copy-on-write reflink. If the platform does not support copy-on-write, then a fallback copy mechanism is used.
* fs.constants.COPYFILE\_FICLONE\_FORCE: The copy operation will attempt to create a copy-on-write reflink. If the platform does not support copy-on-write, then the operation will fail.

# Readline

|  |
| --- |
| Event: 'close'  Event: 'line'  Event: 'pause'  Event: 'resume'  Event: 'SIGCONT'  Event: 'SIGINT'  Event: 'SIGTSTP'   * rl.close() * rl.pause() * rl.prompt([preserveCursor]) * rl.question(query, callback) * rl.resume() * rl.setPrompt(prompt) * rl.getPrompt() * rl.write(data[, key])   rl.write('Delete this!');  // Simulate Ctrl+U to delete the line written previously  rl.write(null, { ctrl: true, name: 'u' });   * data [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#String_type) * key [<Object>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object)   + ctrl [<boolean>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#Boolean_type) true to indicate the **Ctrl** key.   + meta [<boolean>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#Boolean_type) true to indicate the **Meta** key.   + shift [<boolean>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#Boolean_type) true to indicate the **Shift** key.   + name [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#String_type) The name of the a key. * rl.line * rl.cursor * rl.getCursorPos() * readline.clearLine(stream, dir[, callback]) * readline.clearScreenDown(stream[, callback]) * readline.createInterface(options)   Use of the completer function   * readline.cursorTo(stream, x[, y][, callback]) * readline.emitKeypressEvents(stream[, interface]) * readline.moveCursor(stream, dx, dy[, callback])   rl.on('SIGINT', () => {    rl.question('Are you sure you want to exit? ', (answer) => {      if (answer.match(/^y(es)?$/i)) rl.pause();    });  });  /\* Khi Ctrl C \*/ |

const readline = require('readline');

const rl = readline.createInterface({

    input: process.stdin, // process.\_ là const của nodejs

    output: process.stdout

});

let num1 = Math.floor((Math.random() \* 10) + 1);

let num2 = Math.floor((Math.random() \* 10) + 1);

let answer = num1 + num2;

/\* Cách 1 : No loop \*/

rl.question(`${num1} + ${num2} = ?\n`, function(input) {

    if(answer === parseInt(input.trim())) {

        console.log("Bạn đã trả lời đúng");

    } else {

        console.log("Bạn đã trả lời sai");

    }

    rl.close(); // Nếu không close => chạy mãi

});

rl.on("close", () => {

    console.log("Kích hoạt khi readline close()");

})

/\* Cách 2 : Looping \*/

rl.setPrompt(`${num1} + ${num2} = ?\n`);

rl.prompt();

rl.on("line", input => {

    if(parseInt(input.trim()) === answer) {

        console.log("Bạn đã trả lời đúng");

        rl.close();

    } else {

        rl.setPrompt("Lại sai rồi thử lại đi\n");

        rl.prompt();

    }

})

# File System

## Write File

const fs = require("fs");

/\*

Write File : tạo file mỗi ^~ lần chạy

- Tự tạo nếu không có file

\*/

fs.mkdir("./names", err => {

    if(err) throw err;

});

fs.writeFile('./names/filename.txt', 'Hello content!', err => {

    if (err) throw err;

    console.log('Saved!');

});

/\*

Append File : ghi vào file cũ mỗi ^~ lần chạy

- Tạo nếu không có file

\*/

const names = ["Adam", "Eva", "David", "John", "Danny", "Lamb", "Jose"];

names.forEach(name => {

    fs.appendFile("welcome\_student.txt",

        `Welcome to the holiday, Mr.${name}\n`,

        err => {

            if(err) {

                console.log(err);

                return;

            }

        console.log("đã tạo file thành công");

    })

});

/\*

    Write File : tạo file mỗi ^~ lần chạy

    r+ open the file for reading and writing

    w+ open the file for reading and writing, positioning the stream at the beginning of the file. The file is created if not existing

    a open the file for writing, positioning the stream at the end of the file. The file is created if not existing

    a+ open the file for reading and writing, positioning the stream at the end of the file. The file is created if not existing

\*/

fs.open('filename.txt', 'w', function (err, file) {

    if (err) throw err;

});

/\* DELETE \*/

fs.unlink('asd.txt', function (err) {

    if (err) console.log(err.code);

    console.log('File deleted!');

});

/\* JSON

{

    errno: -4058,

    code: 'ENOENT',

    syscall: 'unlink',

    path: 'C:\\Users\\ilaptop\\Desktop\\learn\_nodejs\\asd.txt'

}

\*/

fs.rename('newName.txt', 'welcome\_student.txt', function (err) {

    if (err) throw err;

    console.log('File Renamed!');

});

/\* JSON

{

  errno: -4058,

  code: 'ENOENT',

  syscall: 'rename',

  path: 'C:\\Users\\ilaptop\\Desktop\\learn\_nodejs\\oldName.txt',

  dest: 'C:\\Users\\ilaptop\\Desktop\\learn\_nodejs\\newName.txt'

}

\*/

## Read File

/\* Read File \*/

fs.readFile("welcome\_student.txt", {encoding: "utf-8"}, (err, data) => {

    if(err) throw err;

    console.log(data);

});

## Create / Read Folder

/\* Create Folder \*/

fs.mkdir("student", err => {

    if(err) throw err;

    console.log("New folder !");

});

/\* Remove Folder \*/

fs.rmdir("student", err => {

    if(err) throw err;

    console.log("Delete folder !");

});

/\* Read Folder \*/

const dirName = "./names";

fs.readdir(dirName, (err, fileNames) => {

    if(err) {

        console.log(err);

        throw err;

    }

    fileNames.forEach(name => {

        fs.readFile(`${dirName}/${name}`, "utf-8", (err, data) => {

            if(err) throw err;

            console.log(`From File : ${name}`);

            console.log(data);

        })

    })

})

## Read / Write Stream

Lý do dùng stream : vì nếu file quá lớn(10Gbs) thì buffer không thể nào đủ để đọc hay ghi

Với Stream : file sẽ được split thành ~ chunk nhỏ fit với buffer

const readStream = fs.createReadStream("./welcome\_student.txt", "utf-8");

const writeStream = fs.createWriteStream("./write\_big.txt.gz");

readStream.on("data", chunk => {

    console.log(chunk);

    writeStream.write(chunk, err => {

        if(err) {

            console.log(err);

            throw err;

        }

    })

})

setTimeout(() => {

    readStream.close();

}, 1000);

writeStream.on("drain", () => {

    console.log("hoàn thành việc ghi");

});

readStream.on("close", () => {

    console.log("đã đóng file");

})

/\* Dùng Pipe \*/

readStream.pipe(writeStream);

/\* Zipping \*/

const zlib = require("zlib")

const gzip = zlib.createGzip();

readStream.pipe(gzip).pipe(writeStream);

/\* Unzip \*/

const readStream = fs.createReadStream("./write\_big.txt.gz");

const writeStream = fs.createWriteStream("./write\_big.txt");

const gunzip = zlib.createUnzip();

readStream.pipe(gunzip).pipe(writeStream);

# NPM

{

  "name": "test-project",

  "version": "1.0.0",

  "description": "A Vue.js project",

  "main": "app.js",

  "private": true,

  "scripts": {

    "dev": "webpack-dev-server --inline --progress --config build/webpack.dev.conf.js",

    "start": "npm run dev",

    "unit": "jest --config test/unit/jest.conf.js --coverage",

    "test": "npm run unit",

    "lint": "eslint --ext .js,.vue src test/unit",

    "build": "node build/build.js"

  },

  "dependencies": {

    "vue": "^2.5.2"

  },

  "devDependencies": {

    "file-loader": "^1.1.4",

    "friendly-errors-webpack-plugin": "^1.6.1",

    "html-webpack-plugin": "^2.30.1",

  },

  "engines": {

    "node": ">= 6.0.0",

    "npm": ">= 3.0.0"

  },

  "browserslist": ["> 1%", "last 2 versions", "not ie <= 8"],

  "author": "",

  "license": "ISC"

}

# Environment Properties

console.log(process.env);

/\*

{

    ALLUSERSPROFILE: 'C:\\ProgramData',

    APPDATA: 'C:\\Users\\ilaptop\\AppData\\Roaming',

    COMPUTERNAME: 'SE140344',

    ComSpec: 'C:\\WINDOWS\\system32\\cmd.exe',

    HOMEDRIVE: 'C:',

    HOMEPATH: '\\Users\\ilaptop',

    JAVA\_HOME: 'C:\\Program Files\\Java\\jdk-11.0.3',

    OS: 'Windows\_NT',

    PATHEXT: '.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.CPL',

    PROCESSOR\_ARCHITECTURE: 'AMD64',

    PROCESSOR\_IDENTIFIER: 'Intel64 Family 6 Model 60 Stepping 3, GenuineIntel',

    PROCESSOR\_LEVEL: '6',

    PROCESSOR\_REVISION: '3c03',

    ProgramFiles: 'C:\\Program Files',

    SESSIONNAME: 'Console',

    SystemDrive: 'C:',

    SystemRoot: 'C:\\WINDOWS',

    TEMP: 'C:\\Users\\ilaptop\\AppData\\Local\\Temp',

    TMP: 'C:\\Users\\ilaptop\\AppData\\Local\\Temp',

    USERDOMAIN: 'SE140344',

    USERDOMAIN\_ROAMINGPROFILE: 'SE140344',

    USERNAME: 'ilaptop',

    USERPROFILE: 'C:\\Users\\ilaptop',

    windir: 'C:\\WINDOWS',

}

\*/

# Express

const fs = require("fs")

const express = require("express")

const path = require("path")

const bodyParser = require("body-parser");

const app = express();

app.use("/public", express.static(path.join(\_\_dirname, "static")))

app.use(bodyParser.urlencoded({extended: false})) // nếu muốn nhận xxx-url-encoded contentType

app.use(bodyParser.json()) // nếu muốn json-dataType

app.get("/", (req, res) => {

    res.status(200);

    res.sendFile(path.join(\_\_dirname, "static", "index.html"));

})

app.post('/create',  (req, res) => {

    console.log(req.body);

    res.sendStatus(204);

});

app.listen(3000, () => {

    console.log("Server is running on port 3000");

})

What’s in request

/\*

    httpVersion: '1.1',

    complete: false,

    headers: {

        host: 'localhost:8080',

        connection: 'keep-alive',

        'upgrade-insecure-requests': '1',

        'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.66 Safari/537.36',

        accept: 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng',

        'sec-fetch-site': 'none',

        'sec-fetch-mode': 'navigate',

        'sec-fetch-user': '?1',

        'sec-fetch-dest': 'document',

        'accept-encoding': 'gzip, deflate,

        'accept-language': 'vi-VN,vi;q=0.9,en-US;q=0.8,en;q=0.7',

        'if-none-match': 'W/"329-u3tNsDsO3mAYxqOztCtVUrJkVZc"'

    }

    url: '/',

    method: 'GET'

    client: Socket {

        connecting: false,

        \_hadError: false,

        \_parent: null,

        \_host: null,

        \_readableState: ReadableState {

        objectMode: false,

        highWaterMark: 16384,

        buffer: BufferList { head: null, tail: null, length: 0 },

        length: 0,

        pipes: null,

        pipesCount: 0,

        flowing: true,

        ended: false,

        endEmitted: false,

        reading: true,

        sync: false,

        needReadable: true,

        emittedReadable: false,

        readableListening: false,

        resumeScheduled: false,

        emitClose: false,

        autoDestroy: false,

        destroyed: false,

        defaultEncoding: 'utf8',

        awaitDrainWriters: null,

        multiAwaitDrain: false,

        readingMore: false,

        decoder: null,

        encoding: null,

        [Symbol(kPaused)]: false

    },

    readable: true,

\*/

# Global

|  |  |
| --- | --- |
| * clearImmediate(immediateObject) * clearInterval(intervalObject) * clearTimeout(timeoutObject) * console * Event * EventTarget * exports * global * MessageChannel * MessageEvent * MessagePort | * module * process * queueMicrotask(callback) * require() * setImmediate(callback[, ...args]) * setInterval(callback, delay[, ...args]) * setTimeout(callback, delay[, ...args]) * TextDecoder * TextEncoder * URL * URLSearchParams * WebAssembly |

# Global Object

Không phải của Global mà chỉ là ^~ module đều có

* \_\_dirname
* \_\_filename
* exports
* module
* require()

# Module

## Module wrapper

(function(exports, require, module, \_\_filename, \_\_dirname) {

    // Module code actually lives in here

});

## Require main

console.log(require.main);

Module {

    id: '.',

    path: '/absolute/path/to',

    exports: {},

    filename: '/absolute/path/to/entry.js',

    loaded: false,

    children: [],

    paths: [

        '/absolute/path/to/node\_modules',

        '/absolute/path/node\_modules',

        '/absolute/node\_modules',

        '/node\_modules'

    ]

}

# Path

|  |
| --- |
| * path.basename(path[, ext]) * path.delimiter * ; for Windows * : for POSIX * path.dirname(path) * path.extname(path) * path.format(pathObject) * path.isAbsolute(path) * path.join([...paths]) * path.normalize(path) * path.parse(path) * path.posix * path.relative(from, to) * path.resolve([...paths]) * path.sep * path.toNamespacedPath(path) * path.win32 |

# String Decoder

|  |
| --- |
| Class: StringDecoder   * new StringDecoder([encoding]) * stringDecoder.end([buffer]) * stringDecoder.write(buffer) |

const { StringDecoder } = require('string\_decoder');

const decoder = new StringDecoder('utf8');

const cent = Buffer.from([0xC2, 0xA2]);

console.log(decoder.write(cent));

const euro = Buffer.from([0xE2, 0x82, 0xAC]);

console.log(decoder.write(euro));

# URL

|  |
| --- |
| Class: URL  new URL(input[, base])   * url.hash * url.host * url.hostname * url.href * url.origin * url.password * url.pathname * url.port * url.protocol * Special schemes * url.search * url.searchParams * url.username * url.toString() * url.toJSON() |

const myURL = new URL('https://example.org/foo#bar');

console.log(myURL.hash);

// Prints #bar

# Utils

|  |
| --- |
| util.types   * util.types.isAnyArrayBuffer(value) * util.types.isArrayBufferView(value) * util.types.isArgumentsObject(value) * util.types.isArrayBuffer(value) * util.types.isAsyncFunction(value) * util.types.isBigInt64Array(value) * util.types.isBigUint64Array(value) * util.types.isBooleanObject(value) * util.types.isBoxedPrimitive(value) * util.types.isDataView(value) * util.types.isDate(value) * util.types.isExternal(value) * util.types.isFloat32Array(value) * util.types.isFloat64Array(value) * util.types.isGeneratorFunction(value) * util.types.isGeneratorObject(value) * util.types.isInt8Array(value) * util.types.isInt16Array(value) * util.types.isInt32Array(value) * util.types.isMap(value) * util.types.isMapIterator(value) * util.types.isModuleNamespaceObject(value) * util.types.isNativeError(value) * util.types.isNumberObject(value) * util.types.isPromise(value) * util.types.isProxy(value) * util.types.isRegExp(value) * util.types.isSet(value) * util.types.isSetIterator(value) * util.types.isSharedArrayBuffer(value) * util.types.isStringObject(value) * util.types.isSymbolObject(value) * util.types.isTypedArray(value) * util.types.isUint8Array(value) * util.types.isUint8ClampedArray(value) * util.types.isUint16Array(value) * util.types.isUint32Array(value) * util.types.isWeakMap(value) * util.types.isWeakSet(value) * util.types.isWebAssemblyCompiledModule(value) |

util.types.isDate(new Date());  // Returns true

Còn nữa . . . : Performance hook, timers, trace event, zlib

# Express Framework

## express()

là built-in middleware duyệt JSON payload ở request (dựa trên body-parser)

|  |
| --- |
| Methods |
| * express.json() * express.raw() * express.Router() * express.static() * express.text() * express.urlencoded() |

## Application

|  |  |
| --- | --- |
| Properties | Events |
| * app.locals * app.mountpath   app.locals.title = 'My App'  app.locals.strftime = require('strftime')  app.locals.email = 'me@myapp.com'  console.dir(app.locals.title)  // => 'My App'  console.dir(app.locals.email)  // => 'me@myapp.com'  /\*    Mount Path  \*/  admin.get('/', function (req, res) {    console.log(admin.mountpath) // /admin    res.send('Admin Homepage')  })  app.use('/admin', admin) // mount the sub app | * Mount   admin.on('mount', function (parent) {    console.log('Admin Mounted')    console.log(parent) // refers to the parent app  })  admin.get('/', function (req, res) {    res.send('Admin Homepage')  })  app.use('/admin', admin) |
| Methods | |
| * app.**all**() * app.delete() * app.disable() * app.disabled() * app.enable() * app.enabled() * app.engine() * app.**get**() * app.listen() * app.METHOD() * app.param() * app.path() * app.**post**() * app.**put**() * app.render() * app.route() * app.set()   app.use() | All  app.all('/api/\*', requireAuthentication)  Engine  app.engine('pug', require('pug').\_\_express)  Param  app.param(['id', 'page'], function (req, res, next, value) {    console.log('CALLED ONLY ONCE with', value)    next()  })  app.get('/user/:id/:page', function (req, res, next) {    console.log('although this matches')    next()  })  app.get('/user/:id/:page', function (req, res) {    console.log('and this matches too')    res.end()  })  // CALLED ONLY ONCE with 42  // CALLED ONLY ONCE with 3  // although this matches  // and this matches too  Path  var app = express()  var blog = express()  var blogAdmin = express()  app.use('/blog', blog)  blog.use('/admin', blogAdmin)  console.dir(app.path()) // ''  console.dir(blog.path()) // '/blog'  console.dir(blogAdmin.path()) // '/blog/admin' |
|  |  |

## Request

|  |  |
| --- | --- |
| Prop | Methods |
| * req.app * req.baseUrl * req.body * req.cookies * req.fresh * req.hostname * req.ip * req.ips * req.method * req.originalUrl * req.params * req.path * req.protocol * req.query * req.route * req.secure * req.signedCookies * req.stale * req.subdomains * req.xhr | * req.accepts() * req.acceptsCharsets() * req.acceptsEncodings() * req.acceptsLanguages() * req.get() * req.is() * req.param() * req.range() |

app.get("/student/index/:id", (req, res) => {

  console.log(`App : ${req.app}`);

  console.log(`baseURL : ${req.baseUrl}`);

  console.log(`cookie : ${req.cookies}`);

  console.log(`fresh : ${req.fresh}`);

  console.log(`hostname : ${req.hostname}`);

  console.log(`Ip : ${req.ip}`);

  console.log(`Ips : ${req.ips}`);

  console.log(`Method : ${req.method}`);

  console.log(`originalURL : ${req.originalUrl}`);

  console.log(`params : ${req.params.id}`);

  console.log(`query : ${req.query.limit}`);

  /\*

    App : function(req, res, next) {

        app.handle(req, res, next);

      }

    baseURL :

    cookie : undefined

    fresh : false

    hostname : localhost

    Ip : ::1

    Ips :

    Method : GET

    originalURL : /student/index/1?limit=3000

    params : 1

    query : 3000

  \*/

  res.send({code: "done"});

})

Response

|  |  |
| --- | --- |
| Prop | Methods |
| res.app  res.headersSent  res.locals | res.append()  res.attachment()  res.cookie()  res.clearCookie()  res.download()  res.end()  res.format()  res.get()  res.json()  res.jsonp()  res.links()  res.location()  res.redirect()  res.render()  res.send()  res.sendFile()  res.sendStatus()  res.set()  res.status()  res.type()  res.vary() |
| res.status(500).json({ error: 'message' })  res.get('Content-Type') // getFromHeader  // => "text/plain"  res.links({    next: 'http://api.example.com/users?page=2',    last: 'http://api.example.com/users?page=5'  })  res.location('/foo/bar')  res.location('http://example.com')  res.type('.html')  // => 'text/html'  res.type('html')  // => 'text/html'  res.type('json')  // => 'application/json'  res.type('application/json')  // => 'application/json'  res.type('png')  // => 'image/png'  res.cookie('rememberme', '1', { expires: new Date(Date.now() + 900000), httpOnly: true })  res    .status(201)    .cookie('access\_token', 'Bearer ' + token, {      expires: new Date(Date.now() + 8 \* 3600000) // cookie will be removed after 8 hours    })    .cookie('test', 'test')    .redirect(301, '/admin')  res.download('/report-12345.pdf', 'report.pdf', function (err) {    if (err) {      // Handle error, but keep in mind the response may be partially-sent      // so check res.headersSent    } else {      // decrement a download credit, etc.    }  }) | |

Router

|  |
| --- |
| Methods |
| router.all()  router.METHOD()  router.param()  router.route()  router.use()  router.use(express.static(path.join(\_\_dirname, 'public')))  router.use(express.static(path.join(\_\_dirname, 'public')))  router.use(express.static(path.join(\_\_dirname, 'files')))  router.use(express.static(path.join(\_\_dirname, 'uploads')))  router.use('/bar', function (req, res, next) {    // ... maybe some additional /bar logging ...    next()  }) |