Взаємодія з файловою системою. Модуль **fs**

**Операції з файлами**

Посилання

<https://www.w3schools.com/nodejs/nodejs_filesystem.asp>

**Зчитування даних з файлу**

|  |  |
| --- | --- |
| **Асинхронна версія (неблокуючий підхід)**  **fs.readFile('шлях\_до\_файлу', function(err, data)**  //callback    . . . });  Для перевірки на помилку використовуємо об’єкт **err.** Якщо при зчитуванні файлу сталася помилка, то перший параметр **err** містить об’єкт відповідної виключної ситуації. Якщо помилки немає, то **err** містить null, а параметр **data** містить зчитані дані. | var http = require('http'); var fs = require('fs'); http.createServer(function (req, res) { **fs.readFile('demofile1.html', function(err, data) {**  **// ---- ЯКЩО СТАЛАСЯ ПОМИЛКА ----**  **if(err){**  console.error(err);  res.statusCode = 500;  res.end("На сервері помилка!");  return;  **}**  **//------ ЯКЩО ПОМИЛКИ НЕ БУЛО -----**     res.writeHead(200, {'Content-Type': 'text/html'});     res.write(data);     res.end();   }); }).listen(3000); |
| **Синхронна версія (блокуючий підхід)**  змінна = **fs.readFileSync**('шлях\_до\_файлу', 'кодування');  Для перевірки на помилку використовується **try-catch** | var http = require('http'); var fs = require('fs'); http.createServer(function (req, res) {  var info;    if(req.url == '/'){  try{  info = **fs.readFileSync**("hello.txt", "utf8");  }catch(err){  console.error(err);  res.statusCode = 500;  res.end("На сервері помилка!");  return;  }    res.end(info);  }else{  console.error(“The path is wrong”);  res.statusCode = 404;  res.end("Невірний маршрут!");  return;  }  }).listen(3000); |

**Створення файлів**

|  |  |
| --- | --- |
| **Дописування в існуючий файл (якщо не існує, то буде створено)**  fs.appendFile(файл, контент, function (err) { //callback    . . . });  1) якщо файл існує, то будемо дописувати в кінець;  2) якщо файлу не існує, то файл буде створено;  3) якщо сталася помилка, то буде викликано функцію-обробник, де **err** містить об’єкт помилки; якщо помилки не було, то **err** дорівнює null.  **Синхронний аналог**  fs.appendFileSync("hello.txt", "Привет ми ми ми!"); | var fs = require('fs');  fs.appendFile('mynewfile1.txt', 'Hello content!', function (err) {   if (err) throw err;   console.log('Saved!'); }); |
| **Створення нового файлу**  **fs.open**('шлях\_до\_файлу', 'w', function (err, file) **{**  **обробка помилок**  **запис у файл**  **});** | var fs = require('fs');  fs.open('mynewfile2.txt', 'w', function (err, file) {   if (err) throw err;   console.log('Saved!'); }); |
| **Перезапис файлу (якщо не існувало, то буде створено)**  **fs.writeFile**('шлях\_до\_файлу', 'контент', function (err) { //callback    . . . });  **Синхронний аналог**  fs.writeFileSync("hello.txt", "Привет ми ми ми!") | var fs = require('fs');  fs.writefile('mynewfile3.txt', 'Hello content!', function (err) {   if (err) throw err;   console.log('Saved!'); }); |
|  |  |

**Оновлення вмісту файлів**

|  |  |
| --- | --- |
| **Дописування в існуючий файл (якщо не існує, то буде створено)**  fs.appendFile(файл, контент, function (err) { //callback    . . . });  1) якщо файл існує, то будемо дописувати в кінець;  2) якщо файлу не існує, то файл буде створено;  3) якщо сталася помилка, то буде викликано функцію-обробник, де **err** містить об’єкт помилки; якщо помилки не було, то **err** дорівнює null. | var fs = require('fs');  fs.appendFile('mynewfile1.txt', 'Hello content!', function (err) {   if (err) throw err;   console.log('Saved!'); }); |
| **Перезапис файлу (якщо не існувало, то буде створено)**  **fs.writeFile**('шлях\_до\_файлу', 'контент', function (err) { //callback    . . . }**);** | var fs = require('fs');  fs.writeFile('mynewfile3.txt', 'Hello content!', function (err) {   if (err) throw err;   console.log('Saved!'); }); |

**Видалення файлів**

|  |  |
| --- | --- |
| **fs.unlink**('шлях\_до\_файлу', function (err) { //callback    . . . }); | var fs = require('fs');  fs.unlink('mynewfile2.txt', function (err) {   if (err) throw err;   console.log('File deleted!'); }); |
|  |  |

**Перейменування файлів**

|  |  |
| --- | --- |
| **fs.rename**('**шлях\_до\_файлу**', 'новий\_шлях',  function (err) { //callback    . . . }); | var fs = require('fs');  fs.rename('mynewfile1.txt', 'myrenamedfile.txt', function (err) {   if (err) throw err;   console.log('File Renamed!'); }); |
|  |  |
|  |  |

Робота з директоріями

|  |  |
| --- | --- |
| Створення нової директорії |  |
| **fs.mkdirSync(path[, options])** ----------------fs.mkdir(path[, options], callback)  * path [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#String_type) | [<Buffer>](https://nodejs.org/api/buffer.html#buffer_class_buffer) | [<URL>](https://nodejs.org/api/url.html#url_the_whatwg_url_api) * options [<Object>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object) | [<integer>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#Number_type)   + recursive [<boolean>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#Boolean_type) **Default:** false   + mode [<integer>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#Number_type) Not supported on Windows. **Default:** 0o777. * callback [<Function>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function)   + err [<Error>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Error) | // Creates /tmp/a/apple, regardless of whether `/tmp` and /tmp/a exist.  fs.mkdir('/tmp/a/apple', { recursive: true }, (err) => {  if (err) throw err;  }); |
| Видалення директоріїfs.rmdirSync(path)--------------------fs.rmdir(path, callback)  * path [<string>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#String_type) | [<Buffer>](https://nodejs.org/api/buffer.html#buffer_class_buffer) | [<URL>](https://nodejs.org/api/url.html#url_the_whatwg_url_api) * callback [<Function>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function)   + err [<Error>](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Error)   Asynchronous [rmdir(2)](http://man7.org/linux/man-pages/man2/rmdir.2.html). No arguments other than a possible exception are given to the completion callback.  Using fs.rmdir() on a file (not a directory) results in an ENOENT error on Windows and an ENOTDIR error on POSIX. | fs.rmdir ('/tmp/a/apple', (err) => {  if (err) throw err;  }); |

Завантаження файлів на сервер.

Модуль Formidable

Модуль formidable, express-formidable використовується при завантаженні файлів на сервер.

Посилання:

<https://www.npmjs.com/package/formidable>

<https://www.npmjs.com/package/express-formidable>

|  |
| --- |
| Код на сервері |
| var http = require('http');  //- - - - Підключення модуля - - - -  var formidable = require('formidable'); **var fs = require('fs');**  http.createServer(function (req, res) {   if (req.url == '/fileupload') {  //- - - - Створення об’єкта IncomingForm - - - -     var form = new formidable.IncomingForm();  //- - - - парсинг інформації - - - - ­­    form.parse(req, function (err, fields, files) { **var oldpath = files.filetoupload.path;**  //- - - - Запис завантаженого файлу у нову позицію - - - - **var newpath = 'C:/Users/Your Name/' + files.filetoupload.name;        fs.rename(oldpath, newpath, function (err) {         if (err) throw err;         res.write('File uploaded and moved!');         res.end();       });**  });   } else {     res.writeHead(200, {'Content-Type': 'text/html'});     res.write('<form action="fileupload" method="post" enctype="multipart/form-data">');     res.write('<input type="file" name="filetoupload"><br>');     res.write('<input type="submit">');     res.write('</form>');     return res.end();   } }).listen(8080); |

Властивості IncomingForm

|  |  |
| --- | --- |
| Sets encoding for incoming form fields. | form.encoding **=** 'utf-8'; |
| Sets the directory for placing file uploads in. You can move them later on using fs.rename(). The default is os.tmpdir(). | form.uploadDir **=** "/my/dir"; |
| If you want the files written to form.uploadDir to include the extensions of the original files, set this property to true. | form.keepExtensions **=** false; |
| Either 'multipart' or 'urlencoded' depending on the incoming request. | form.type |
| Limits the amount of memory all fields together (except files) can allocate in bytes. If this value is exceeded, an 'error' event is emitted. The default size is 20MB. | form.maxFieldsSize **=** 20 **\*** 1024 **\*** 1024; |
| Limits the size of uploaded file. If this value is exceeded, an 'error' event is emitted. The default size is 200MB. | form.maxFileSize **=** 200 **\*** 1024 **\*** 1024; |
| Limits the number of fields that the querystring parser will decode. Defaults to 1000 (0 for unlimited). | form.maxFields **=** 1000; |
| If you want checksums calculated for incoming files, set this to either 'sha1' or 'md5'. | form.hash **=** false; |
| If this option is enabled, when you call form.parse, the files argument will contain arrays of files for inputs which submit multiple files using the HTML5 multiple attribute. | form.multiples **=** false; |
| The amount of bytes received for this form so far. | form.bytesReceived |
| The expected number of bytes in this form | form.bytesExpected |
| Parses an incoming node.js request containing form data. If cb is provided, all fields and files are collected and passed to the callback: | form.parse(request, [cb]); |
|  |  |

Використання express-formidable

|  |
| --- |
|  |
| const express **=** require('express');  const formidableMiddleware **=** require('express-formidable');    var app **=** express();    app.use(formidableMiddleware());    app.post('/upload', (req, res) => {    req.fields; *// contains non-file fields*    req.files; *// contains files*  }) |