TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN ỨNG DỤNG WEB VỚI NODEJS (502070)

LAB6/7 – NODEJS

TDTU

Express database integration (expressjs.com)

- Cassandra
- Couchbase
- CouchDB
- LevelDB
- **MySQL**
- MongoDB
- Neo4j
- Oracle
- PostgreSQL
- Redis
- SQL Server
- SQLite
- Elasticsearch

Express database integration (expressjs.com)

```
$ npm install mysql
```

```
const mysql = require('mysql')
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'dbuser',
  password: 's3kreee7',
  database: 'my_db'
})

connection.connect()

connection.query('SELECT 1 + 1 AS solution', (err, rows, fields) => {
  if (err) throw err

  console.log('The solution is: ', rows[0].solution)
})

connection.end()
```

promise-mysql - npm (npmjs.com)

Connector/Node.js Promise API - MariaDB Knowledge Base

```
var mysql = require('promise-mysql');

var connection;
var arr = [];

mysql.createConnection({
    host: 'host',
    user: 'user',
    password: 'password',
    database: 'database'
}).then(
    function (conn) {
        connection = conn;
        return conn.query('select * from users_groups where user_id=2');
    }
).then(
    function(value) {
        console.log('Initial value : ' + JSON.stringify(value) );
```

bcrypt - npm (npmjs.com)

```
$2<a/b/x/y>$[cost]$[22 character salt][31 character hash]
```

For example, with input password `abc123xyz`, cost `12`, and a random salt, the output of bcrypt is the string

```
$2a$12$R9h/cIPz0gi.URNNX3kh2OPST9/PgBkqquzi.Ss7KIUgO2t0jWMUW
\__/\/ \____________________/\_____________________________/
Alg Cost      Salt                        Hash
```

Where:

- `$2a$`: The hash algorithm identifier (bcrypt)
- `12`: Input cost (2^{12} i.e. 4096 rounds)
- `R9h/cIPz0gi.URNNX3kh2O`: A radix-64 encoding of the input salt
- `PST9/PgBkqquzi.Ss7KIUgO2t0jWMUW`: A radix-64 encoding of the first 23 bytes of the computed 24 byte hash

bcrypt - npm (npmjs.com)

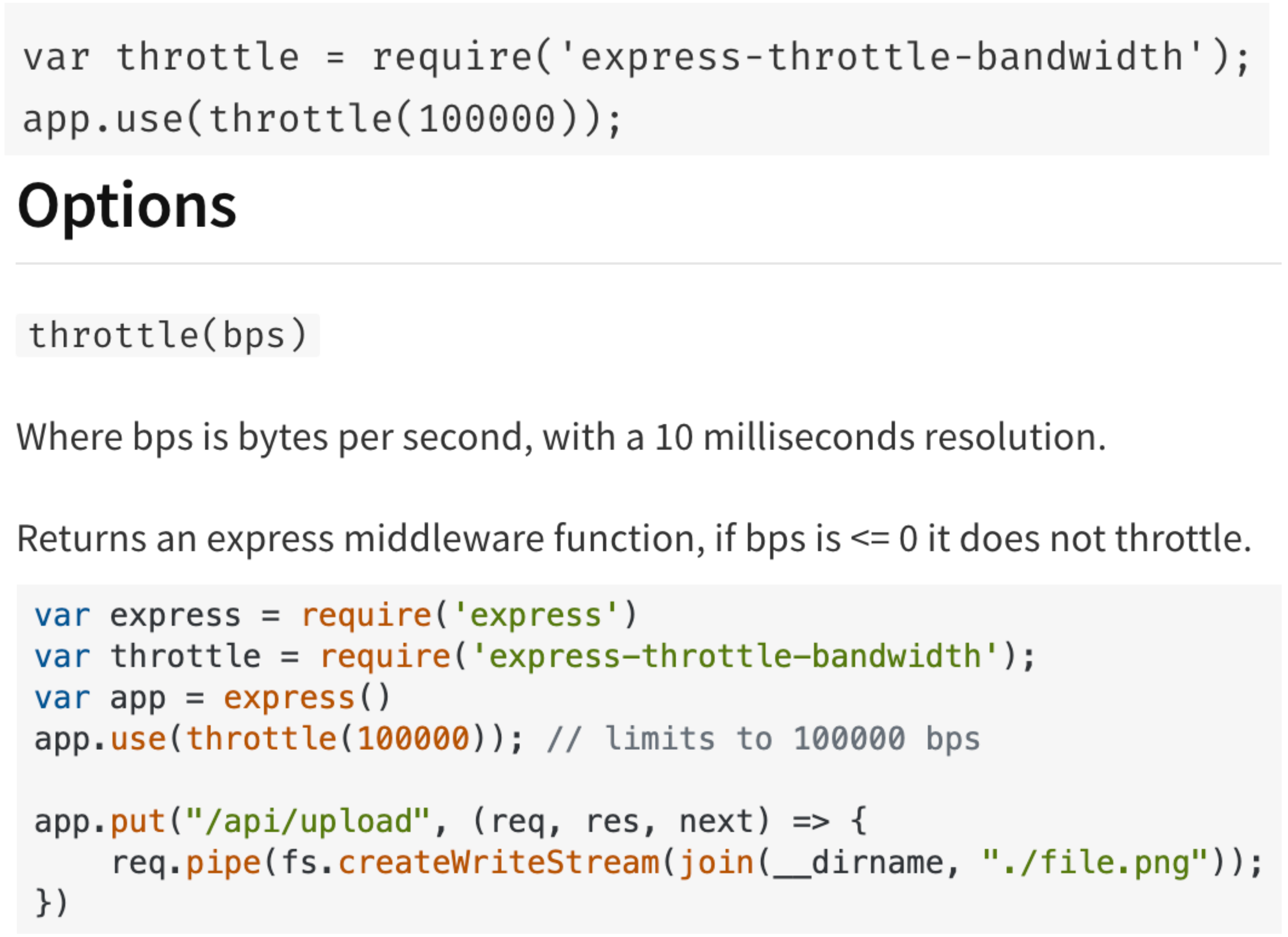
```
const bcrypt = require('bcrypt')

var salt = bcrypt.genSaltSync(10)
var hash = bcrypt.hashSync('B4c0/\/', salt)

// To check a password
var res = bcrypt.compareSync('B4c0/\/', hash)   // true
console.log('equal')
console.log(res)

res = bcrypt.compareSync('not_bacon', hash)     // false
console.log('not equal')
console.log(res)

// Auto-gen a salt and hash
var hash = bcrypt.hashSync('bacon', 8)
console.log(`Auto-gen: ${hash}`)
```

```
bcrypt.genSalt(10, function (err, salt) {
    bcrypt.hash('B4c0/\/', salt, function (err, hash) {

        console.log(hash)

        // To check a password
        bcrypt.compare('B4c0/\/', hash, function (err, res) {
            // res == true
            console.log('equal')
            console.log(res)
        })

        bcrypt.compare('not_bacon', hash, function (err, res) {
            // res == false
            console.log('not equal')
            console.log(res)
        })
    })
})

// Auto-gen a salt and hash
bcrypt.hash('bacon', 8, function (err, hash) {
    console.log(`Auto-gen: ${hash}`)
```

Lab6/7 – Throttle – Limit download

[throttle - npm (npmjs.com)]

[express-throttle-bandwidth - npm (npmjs.com)]



```
var throttle = require('express-throttle-bandwidth');
app.use(throttle(100000));
```

Options

`throttle(bps)`

Where bps is bytes per second, with a 10 milliseconds resolution.

Returns an express middleware function, if bps is <= 0 it does not throttle.

```
var express = require('express')
var throttle = require('express-throttle-bandwidth');
var app = express()
app.use(throttle(100000)); // limits to 100000 bps

app.put("/api/upload", (req, res, next) => {
    req.pipe(fs.createWriteStream(join(__dirname, "./file.png"));
})
```

Express csurf middleware (expressjs.com)

- Cross-site request forgery (CSRF) attacks exploit the fact that users generally trust their browser and visit multiple sites in the same session. In a CSRF attack, script on a malicious site makes requests of another site: if you are logged in on the other site, the malicious site can successfully access secure data from another site.
- To prevent CSRF attacks, you must have a way to make sure a request legitimately came from your website. The way we do this is to pass a unique token to the browser. When the browser then submits a form, the server checks to make sure the token matches.

Express csurf middleware (expressjs.com)

The csurf middleware will handle the token creation and verification for you; all you'll have to do is make sure the token is included in requests to the server. Install the csurf middleware (npm install –save csurf), then link it in and add a token to res.locals:

```
// this must come after we link in cookie-parser and connect-session
app.use(require('csurf')());
app.use(function(req, res, next){
        res.locals._csrfToken = req.csrfToken();
        next();
});
```

Now on all of your forms (and AJAX calls), you'll have to provide a field called _csrf, which must match the generated token.

```
<form action="/newsletter" method="POST">
        <input type="hidden" name="_csrf" value="{{_csrfToken}}">
        Name: <input type="text" name="name"><br>
```

[Content-Disposition - HTTP | MDN (mozilla.org)]

[Express 4.x - API Reference (expressjs.com)]

In a regular HTTP response, the **Content-Disposition** response header is a header indicating if the content is expected to be displayed *inline* in the browser, that is, as a Web page or as part of a Web page, or as an *attachment*, that is downloaded and saved locally.

```
res.attachment()
// Content-Disposition: attachment

res.attachment('path/to/logo.png')
// Content-Disposition: attachment; filename="logo.png"
// Content-Type: image/png
```

```
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.
  }
})
```

PHÁT TRIỂN ỨNG DỤNG WEB VỚI NODEJS

Lab6/7 – Fs Readdir

[File system | Node.js v17.8.0 Documentation (nodejs.org)]

[The Node.js fs module (nodejs.dev)]

[Node.js - File System (tutorialspoint.com)]

```
const fs = require("fs");

fs.readdir("./files", (err, items) => {
  for (const dirent of items) {
    console.log(dirent);
  }
});
```

Get File Information

Syntax

Following is the syntax of the method to get the information about a file −

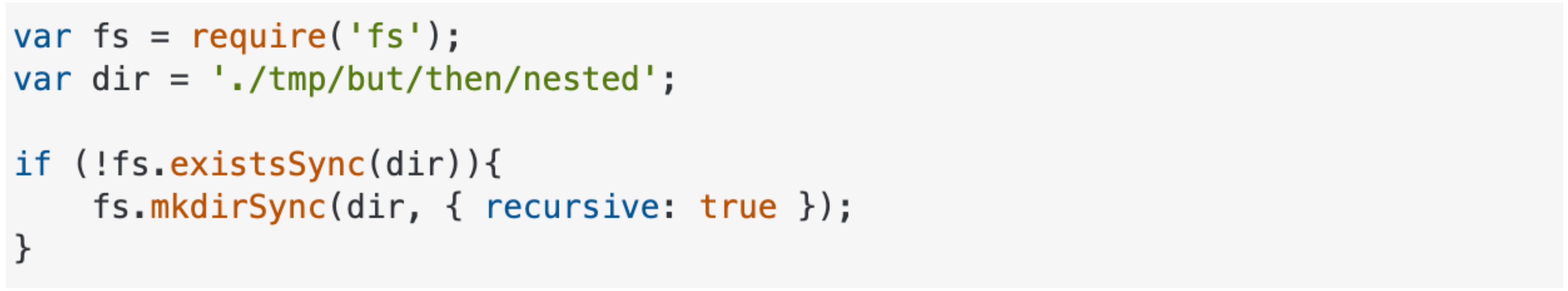
```
fs.stat(path, callback)
```

PHÁT TRIỂN ỨNG DỤNG WEB VỚI NODEJS
Lab6/7 – Fs module

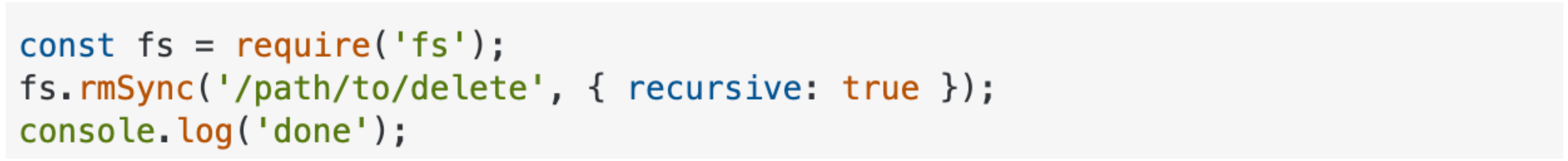
[File system | Node.js v17.8.0 Documentation (nodejs.org)]

[The Node.js fs module (nodejs.dev)]

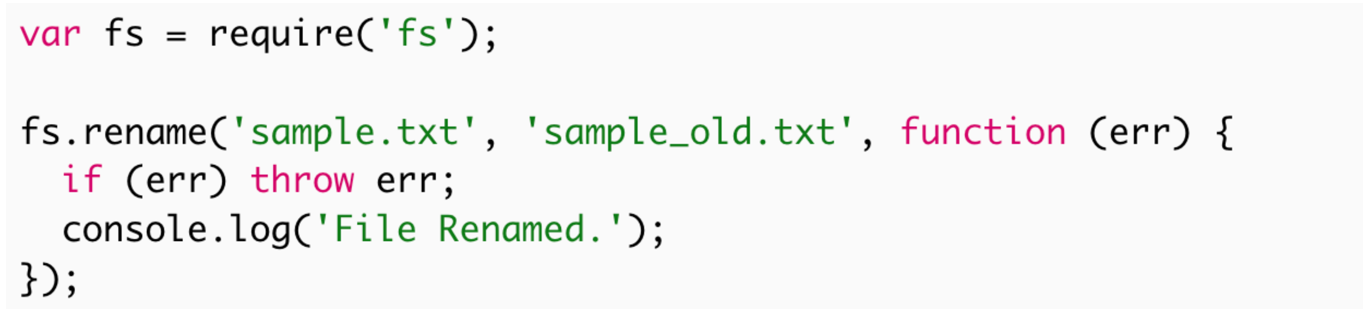
[Node.js - File System (tutorialspoint.com)]



```
var fs = require('fs');
var dir = './tmp/but/then/nested';

if (!fs.existsSync(dir)){
    fs.mkdirSync(dir, { recursive: true });
}
```

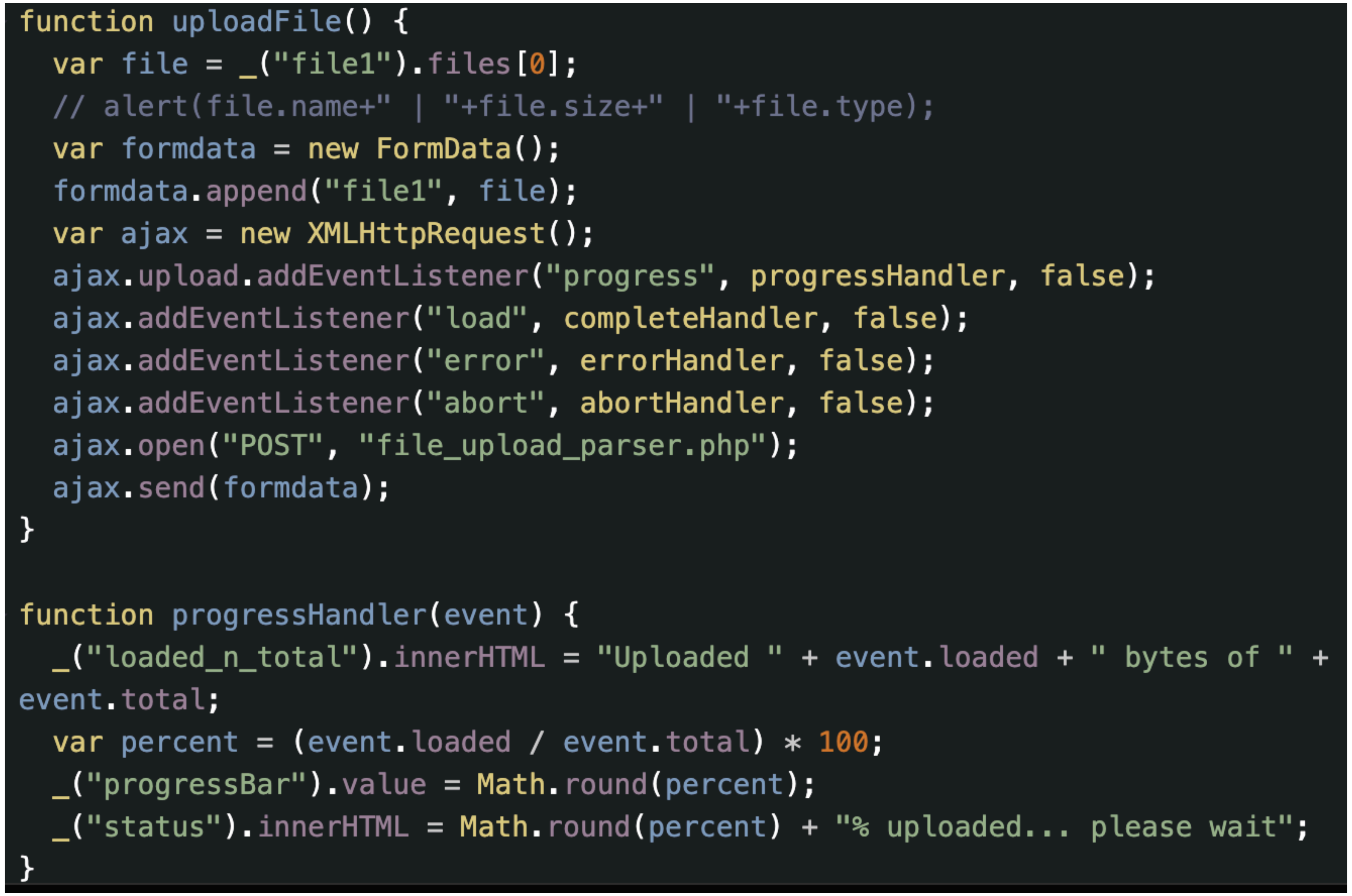


```
const fs = require('fs');
fs.rmSync('/path/to/delete', { recursive: true });
console.log('done');
```



```
var fs = require('fs');

fs.rename('sample.txt', 'sample_old.txt', function (err) {
  if (err) throw err;
  console.log('File Renamed.');
});
```

https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/upload



```
function uploadFile() {
  var file = _("file1").files[0];
  // alert(file.name+" | "+file.size+" | "+file.type);
  var formdata = new FormData();
  formdata.append("file1", file);
  var ajax = new XMLHttpRequest();
  ajax.upload.addEventListener("progress", progressHandler, false);
  ajax.addEventListener("load", completeHandler, false);
  ajax.addEventListener("error", errorHandler, false);
  ajax.addEventListener("abort", abortHandler, false);
  ajax.open("POST", "file_upload_parser.php");
  ajax.send(formdata);
}

function progressHandler(event) {
  _("loaded_n_total").innerHTML = "Uploaded " + event.loaded + " bytes of " +
event.total;
  var percent = (event.loaded / event.total) * 100;
  _("progressBar").value = Math.round(percent);
  _("status").innerHTML = Math.round(percent) + "% uploaded... please wait";
}
```

Lab6/7 – Download

[GitHub - archiverjs/node-archiver: a streaming interface for archive generation]()

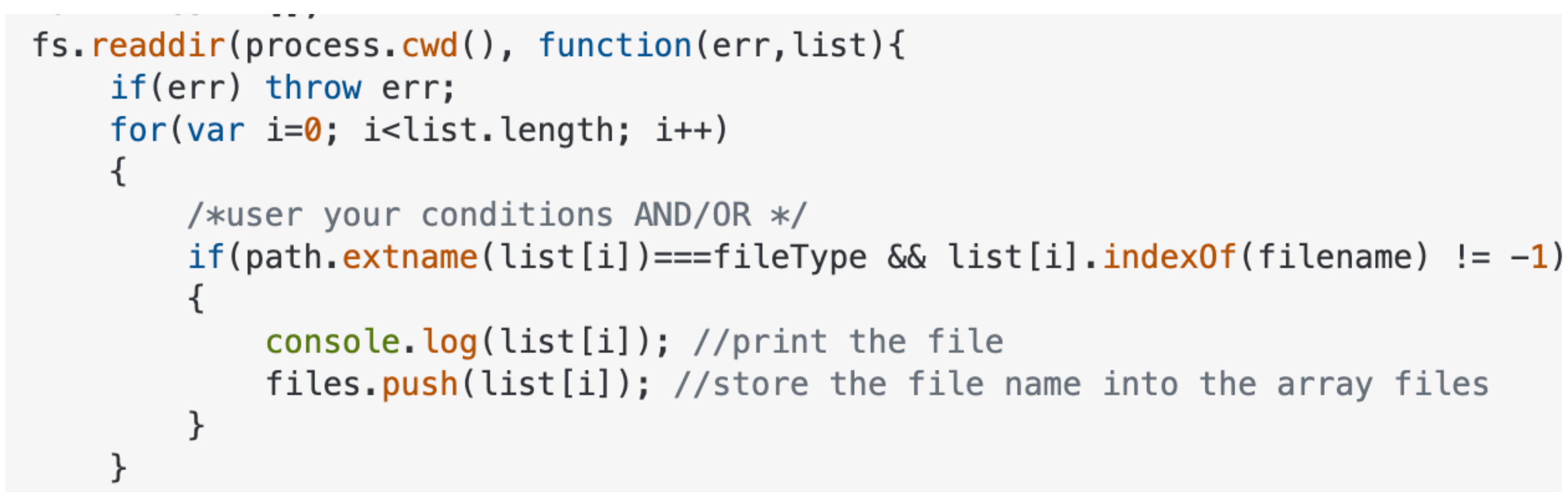
```
var fs = require('fs');
var archiver = require('archiver');

var archive = archiver.create('zip', {});
var output = fs.createWriteStream(__dirname + '/zip_folder.zip');
```

```
archive.pipe(output);

archive
  .directory(__dirname + '/folder_1/folder_2/folder_3/download_folder/zip_folder')
  .finalize();
```

[The Node.js fs module (nodejs.dev)]

[GitHub - isaacs/node-glob: glob functionality for node.js]

[node-find-files - npm (npmjs.com)]



```
fs.readdir(process.cwd(), function(err,list){
    if(err) throw err;
    for(var i=0; i<list.length; i++)
    {
        /*user your conditions AND/OR */
        if(path.extname(list[i])===fileType && list[i].indexOf(filename) != -1)
        {
            console.log(list[i]); //print the file
            files.push(list[i]); //store the file name into the array files
        }
    }
```

Reference links

@syncfusion/ej2-filemanager-node-filesystem - npm (npmjs.com)

GitHub - serverwentdown/file-manager: A basic node.js file manager

GitHub - hiiamrohit/nodeJs-file-upload: File upload in nodeJs with progress bar

nodeJs-file-upload/app.js at master · hiiamrohit/nodeJs-file-upload · GitHub

File Upload Progress bar (codepen.io)

Thank you