

File System

NodeJS

File System

NodeJS implémente la gestion des I/O via le module *fs*.

```
var fs = require("fs")
```

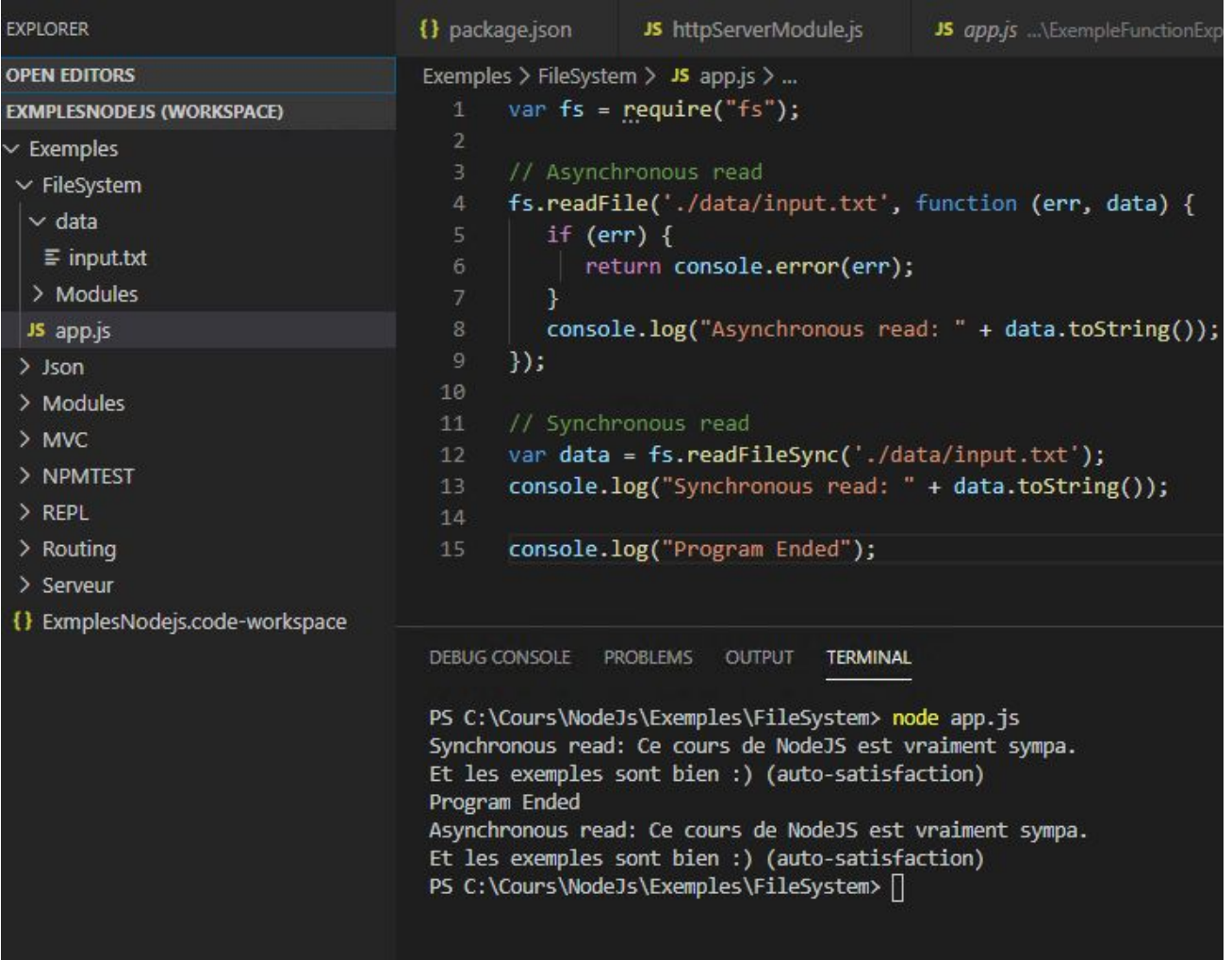
Synchrone et Asynchrone

Nous avons la possibilité de travailler des deux manières avec *fs*

```
fs.readFile(path, callback);
```

Ou

```
Let data = fs.readFileSync(path);
```



The screenshot displays the Visual Studio Code interface. On the left, the Explorer sidebar shows a workspace named 'ExemplesNodejs.code-workspace' with a folder 'Exemples' containing a subfolder 'FileSystem'. Inside 'FileSystem', there is a file 'app.js' which is currently open in the editor. The code in 'app.js' demonstrates both asynchronous and synchronous file reading using the 'fs' module. The asynchronous part uses 'fs.readFile' with a callback function, and the synchronous part uses 'fs.readFileSync'. The terminal at the bottom shows the command 'node app.js' being executed, resulting in two lines of output: 'Synchronous read: Ce cours de NodeJS est vraiment sympa. Et les exemples sont bien :) (auto-satisfaction)' followed by 'Program Ended', and then 'Asynchronous read: Ce cours de NodeJS est vraiment sympa. Et les exemples sont bien :) (auto-satisfaction)'. The prompt 'PS C:\Cours\NodeJs\Exemples\FileSystem>' is visible at the end of the terminal output.

```
package.json  JS httpServerModule.js  JS app.js ...\ExempleFunctionExp

EXPLORER
OPEN EDITORS
EXMPLESNODEJS (WORKSPACE)
  Examples
    FileSystem
      data
        input.txt
      Modules
      JS app.js
      Json
      Modules
      MVC
      NPMTEST
      REPL
      Routing
      Serveur
  ExmplesNodejs.code-workspace

Examples > FileSystem > JS app.js > ...
1  var fs = require("fs");
2
3  // Asynchronous read
4  fs.readFile('./data/input.txt', function (err, data) {
5    if (err) {
6      return console.error(err);
7    }
8    console.log("Asynchronous read: " + data.toString());
9  });
10
11 // Synchronous read
12 var data = fs.readFileSync('./data/input.txt');
13 console.log("Synchronous read: " + data.toString());
14
15 console.log("Program Ended");

DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL
PS C:\Cours\NodeJs\Exemples\FileSystem> node app.js
Synchronous read: Ce cours de NodeJS est vraiment sympa.
Et les exemples sont bien :) (auto-satisfaction)
Program Ended
Asynchronous read: Ce cours de NodeJS est vraiment sympa.
Et les exemples sont bien :) (auto-satisfaction)
PS C:\Cours\NodeJs\Exemples\FileSystem> 
```

File System

Ouverture de fichier

Nous pouvons préciser des *flags* pour l'ouverture du fichier afin de le traiter en lecture seule par exemple

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

Exemples > FileSystem > JS appOpen.js > ...

```
1  var fs = require("fs");
2
3  // Asynchronous - Ouverture du fichier
4  console.log("Nous allons ouvrir le fichier");
5  fs.open('./data/input.txt', 'r+', function(err, fd) {
6      if (err) {
7          return console.error(err);
8      }
9      console.log("Le fichier est ouvert!");
10 });
```

Flags	
r	Ouverture en lecture seule (exception si le fichier n'existe pas)
r+	Ouverture en lecture/écriture (exception si le fichier n'existe pas)
rs	Ouverture en lecture seule en mode synchrone
rs+	Ouverture en lecture/écriture en mode synchrone
w	Ouverture en écriture (le fichier est créé si il n'existe pas ou vidé si il existe)
wx	Ouverture en écriture (le fichier est créé si il n'existe pas ou échec si il existe)
w+	Ouverture en écriture/lecture (le fichier est créé si il n'existe pas ou vidé si il existe)
wx+	Ouverture en écriture/lecture (le fichier est créé si il n'existe pas ou échec si il existe)
a	Permet d'ajouter du contenu à un fichier (créé si le fichier n'existe pas)
ax	Permet d'ajouter du contenu à un fichier (Exception si le fichier n'existe pas)
a+	Permet de lire et d'ajouter du contenu à un fichier (créé si le fichier n'existe pas)
ax+	Permet lire et d'ajouter du contenu à un fichier (Exception si le fichier n'existe pas)

File System

Obtenir des informations sur le fichier

fs nous propose la fonction *stat*

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

Methodes

stats.isFile() renvoie true si c'est un fichier

stats.isDirectory() renvoie true si c'est un dossier

stats.isBlockDevice() renvoie true si c'est un *block device* (cdrom, disquette,...)

stats.isCharacterDevice() renvoie true si c'est une *character device* (port serie, carte son,...)

stats.isSymbolicLink() renvoie true si c'est un lien symbolic (pointer vers un autre fichier ☐ linux)

stats.isSocket() renvoie true si c'est un type socket

```
Exemples > FileSystem > JS appStats.js > ...
1  var fs = require("fs");
2
3  console.log("Récupération des infos du fichier");
4  fs.stat('./data/input.txt', function (err, stats) {
5      if (err) {
6          return console.error(err);
7      }
8      console.log(stats);
9      console.log("Récupération ok!");
10
11     // Check file type
12     console.log("Est-ce un fichier ? " + stats.isFile());
13     console.log("Est-ce un dossier ? " + stats.isDirectory());
14 });
```

DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Cours\NodeJs\Exemples> cd .\FileSystem\

PS C:\Cours\NodeJs\Exemples\FileSystem> node .\appStats.js

Récupération des infos du fichier

```
Stats {
  dev: 4038446957,
  mode: 33206,
  nlink: 1,
  uid: 0,
  gid: 0,
  rdev: 0,
  blksize: 4096,
  ino: 1125899907166597,
  size: 88,
  blocks: 0,
  atimeMs: 1586942568794.74,
  mtimeMs: 1586942568794.74,
  ctimeMs: 1586942568794.74,
  birthtimeMs: 1586942540297.8015,
  atime: 2020-04-15T09:22:48.795Z,
  mtime: 2020-04-15T09:22:48.795Z,
  ctime: 2020-04-15T09:22:48.795Z,
  birthtime: 2020-04-15T09:22:20.298Z
}
```

Récupération ok!

Est-ce un fichier ? true

Est-ce un dossier ? false

File System

Ecrire dans un fichier

Pour écrire (écraser le contenu aussi) d'un fichier :

```
fs.writeFile(filename, data[,options],callback);
```

- Data : string ou buffer
- Options : un objet json

```
{  
  encoding (UTF8),  
  mode (0666),  
  flag (w)  
}
```

Exemples > FileSystem > JS appWrite.js > ...

```
1  var fs = require("fs");  
2  
3  fs.writeFile('./data/Nouveau.txt', 'C'est écrit (Francis Cabrel)', function(err) {  
4    if (err) {  
5      return console.error(err);  
6    }  
7  
8    console.log("Ecriture OK");  
9    console.log("Lecture du fichier");  
10  
11    fs.readFile('./data/Nouveau.txt', function (err, data) {  
12      if (err) {  
13        return console.error(err);  
14      }  
15      console.log("Lecture async: " + data.toString());  
16    });  
17  });
```

DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

```
PS C:\Cours\NodeJs\Exemples\FileSystem> node .\appwrite.js  
Ecriture OK  
Lecture du fichier  
Lecture async: C'est écrit (Francis Cabrel)  
PS C:\Cours\NodeJs\Exemples\FileSystem> []
```

File System

Lire un fichier

Pour lire via un fichier

```
fs.read(fd, buffer, offset, length, position, callback)
```

- fd : la valeur retournée par un open
- Buffer : contiendra les données
- Offset : position dans le buffer pour l'écriture
- Length : nombre de bytes à lire
- Position : int définissant le début de la lecture.
Si null, lecture à la position courante

```
Exemples > FileSystem > JS appReadBuffer.js > fs.open('./data/input.txt', 'r+') callback
1  var fs = require("fs");
2  var buf = Buffer.alloc(1024);
3
4  console.log("Ouverture du fichier existant");
5  fs.open('./data/input.txt', 'r+', function(err, fd) {
6      if (err) {
7          return console.error(err);
8      }
9      console.log("Ouverture ok!");
10     console.log("Lecture");
11
12     fs.read(fd, buf, 0, buf.length, 0, function(err, bytes){
13         if (err){
14             console.log(err);
15         }
16         console.log(bytes + " => Nb bytes lus");
17
18         if(bytes > 0){
19             console.log("-");
20             console.log(buf.slice(0, bytes).toString());
21         }
22     });
23 }
24 };
```

```
DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL
PS C:\Cours\NodeJs\Exemples\FileSystem> node .\appReadBuffer.js
Ouverture du fichier existant
Ouverture ok!
Lecture
88 => Nb bytes lu
-
Ce cours de NodeJS est vraiment sympa.
Et les exemples sont bien :) (auto-satisfaction)
PS C:\Cours\NodeJs\Exemples\FileSystem>
```


File System

Fermer un fichier

```
// Close the opened file.
fs.close(fd, function(err) {
  if (err) {
    console.log(err);
  }
  console.log("Fichier fermé.");
});
```

Créer un dossier

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

console.log("On crée le dossier ./test");
fs.mkdir('./test',function(err) {
  if (err) {
    return console.error(err);
  }
  console.log("Dossier créé!");
});
```

Supprimer un fichier

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

fs.unlink('./data/Nouveau.txt', function(err) {
  if (err) {
    return console.error(err);
  }
  console.log("Fichier supprimé!");
});
```

Parcourir un dossier

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

console.log("lire le dossier ./data/");
fs.readdir("./data",function(err, files) {
  if (err) {
    return console.error(err);
  }
  files.forEach( function (file) {
    console.log( file );
  });
});
```

File System

Supprimer un dossier

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

console.log("supprimer ./test");
fs.rmdir("./test",function(err, files) {
  if (err) {
    return console.error(err);
  }
  console.log("Dossier supprimé!")
});
```