

Modules

NodeJs

Modules

L'utilisation **Module** permet de simplifier l'écriture de code et de le gérer dans votre application.

Normalement, chaque **module** sera écrit dans un fichier séparé.

NodeJS intègre de nombreux **Modules** (Bibliothèques standards) dont les plus important ci-après :

Core Module	Description
http	http module includes classes, methods and events to create Node.js http server.
url	url module includes methods for URL resolution and parsing.
querystring	querystring module includes methods to deal with query string.
path	path module includes methods to deal with file paths.
fs	fs module includes classes, methods, and events to work with file I/O.
util	util module includes utility functions useful for programmers.

Modules

Charger un module

Pour pouvoir utiliser un module dans NodeJS, il faut utiliser la fonction *Require*

```
Var module = require('module_name');
```

```
var http = require('http');  
  
var server = http.createServer(function(req, res){  
    //write code here  
});  
  
server.listen(5000); |
```

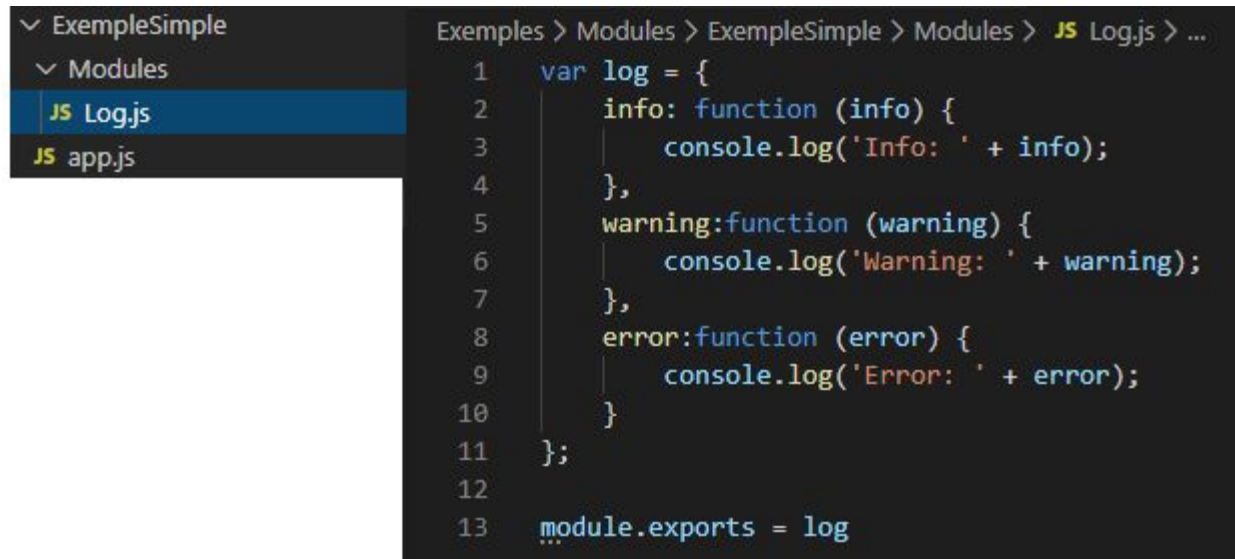
La fonction *require* renvoie un objet qui peut donc être utilisé par la suite dans le script.

Modules

Module local

Un module local est donc un module directement créé dans notre application NodeJs.

Ces modules locaux vont donc être placés dans différents dossier/fichiers pour fournir différentes fonctionnalités récurrentes.



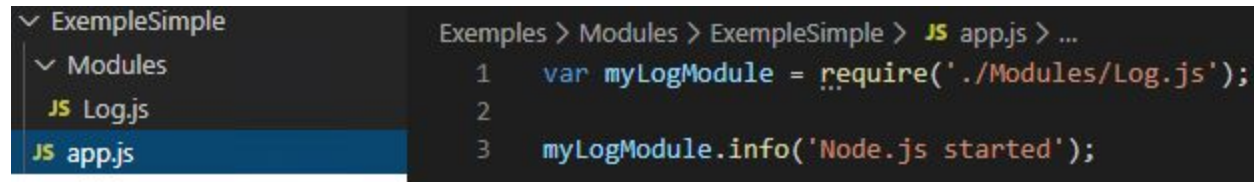
```
1  var log = {
2      info: function (info) {
3          console.log('Info: ' + info);
4      },
5      warning: function (warning) {
6          console.log('Warning: ' + warning);
7      },
8      error: function (error) {
9          console.log('Error: ' + error);
10     }
11 };
12
13 module.exports = log
```

Dans cet exemple, nous déclarons un objet contenant différentes fonctions et c'est via **module.exports** que nous définissons celui-ci comme étant un module.

Modules

Module local

Pour charger le module local dans notre app, nous utilisons la même fonction *require*



```
ExempleSimple
├── Modules
│   ├── Log.js
│   └── app.js
└── ...

Exemples > Modules > ExempleSimple > JS app.js > ...
1  var myLogModule = require('./Modules/Log.js');
2
3  myLogModule.info('Node.js started');
```

Ce qui nous donne le résultat suivant :

```
PS C:\Cours\NodeJs\Exemples\Modules\ExempleSimple> node app.js
Info: Node.js started
PS C:\Cours\NodeJs\Exemples\Modules\ExempleSimple> █
```

Modules

Export Module

Module.exports ou exports est un objet permettant inclus dans n'importe quel fichier js sous NodeJs.

Module est une variable qui représente le module courant et *exports* est un objet qui est exposé en tant que Module.

En résumé, tout ce que vous assignez à Module.exports ou exports sera considéré comme un module par NodeJs.

- Export literals

Nous pouvons exporter directement une valeur.

```
module.exports = 'Hello world'; //ou exports = 'Hello world';
```

- Export Object/function/class

```
module.exports = function (msg) {  
  console.log(msg);  
};
```

Exemples > Modules > ExempleFunctionExport > JS app.js > ...

```
1 var msg = require('./Modules/Log.js');  
2  
3 msg('Hello World');
```

```
module.exports = function (firstName, lastName) {  
  this.firstName = firstName;  
  this.lastName = lastName;  
  this.fullName = function () {  
    return this.firstName + ' ' + this.lastName;  
  }  
}
```

Exemples > Modules > ExempleClassExport > JS app.js > ...

```
1 var person = require('./Modules/Person.js');  
2  
3 var person1 = new person('James', 'Bond');  
4  
5 console.log(person1.fullName());
```