

Node.js

Node.js



Node.js

Node.js

Partie 1



NODE.JS

- > Runtime JavaScript
 - ▶ Créé par Ryan Dahi en 2009
 - ▶ Sous licence MIT
 - ▶ Utilise le moteur V8 (celui de Chrome)
 - ▶ Disponible pour Windows, Linux et MacOS
- > Fonctionnement à base d'événements
- > Adapté pour les applications réseau devant pouvoir monter en charge
- > Permet de monter son propre serveur HTTP
- > Utilisé par : LinkedIn, Paypal, Rakuten, Groupon...
- > Base d'un important écosystème de modules

HELLO WORLD

- > Installer Node.js et s'assurer qu'il peut être lancé en ligne de commande

```
1 Pibook:b3_nodejs william$ node -v
2 v10.12.0
```

- > Créer un fichier **app.js** avec le code suivant :

```
app.js
1 console.log("Bonjour Node.JS");
```

- > Pour exécuter l'application faire en ligne de commande :

```
1 Pibook:hello_world william$ node app.js
2 Bonjour Node.JS
```

Node.js

Node.js

Partie 2



UTILISATION DE MODULES

- > Node.JS dispose d'une large bibliothèque de modules
 - ▶ Simples fichiers à inclure dans un programme
 - ▶ Packages à installer avec la commande **npm**
 - ▶ Modules intégrés à Node.js (comme **readline**, **os**, **fs** ou **http**)
- > readline : module pour effectuer des saisies dans la console
- > os : module pour représenter la machine hôte
- > fs : pour accéder au système de fichiers
- > http : pour traiter des requêtes http

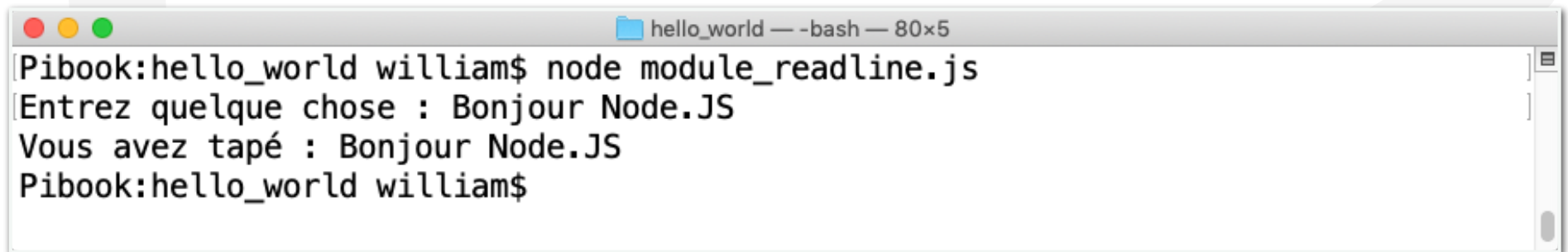
MODULE READLINE

- > Créer le fichier `module_readline.js` :

`module_readline.js`

```
1  const readline = require('readline');
2  const rl = readline.createInterface({
3    input: process.stdin,
4    output: process.stdout
5  });
6
7
8  rl.question('Entrez quelque chose : ', (data) => {
9    console.log("Vous avez tapé : " + data);
10    rl.close();
11  });
```

- > Exécuter en faisant : `node module_readline.js`



A terminal window titled "hello_world — -bash — 80x5" showing the execution of the `module_readline.js` script. The prompt is `Pibook:hello_world william$`. The user enters `node module_readline.js`. The program prompts `Entrez quelque chose :` , the user enters `Bonjour Node.JS`, and the program outputs `Vous avez tapé : Bonjour Node.JS`. The prompt returns to `Pibook:hello_world william$`.

```
Pibook:hello_world william$ node module_readline.js
Entrez quelque chose : Bonjour Node.JS
Vous avez tapé : Bonjour Node.JS
Pibook:hello_world william$
```

- > Attention : ce type de code ne permet pas d'effectuer plus d'une saisie par programme !

MODULE OS

- > Créer le fichier `module_os.js` :

`module_os.js`

```
1  const os = require ('os');  
2  
3  console.log("Architecture du CPU : "+os.arch());  
4  console.log("CPU : "+os.cpus()[0].model);  
5  console.log("Mémoire : "+os.totalmem());  
6  console.log("Dossier utilisateur "+os.homedir());
```

- > Exécuter en faisant : `node module_os.js`

A terminal window titled 'hello_world — -bash — 80x6' showing the execution of the 'node module_os.js' command. The output displays system information: CPU architecture (x64), CPU model (Intel(R) Core(TM) i7-3615QM CPU @ 2.30GHz), total memory (17179869184), and the user's home directory (/Users/william).

```
Pibook:hello_world william$ node module_os.js  
Architecture du CPU : x64  
CPU : Intel(R) Core(TM) i7-3615QM CPU @ 2.30GHz  
Mémoire : 17179869184  
Dossier utilisateur /Users/william  
Pibook:hello_world william$
```

- > Documentation : <https://nodejs.org/api/os.html>

MODULE FS : LIRE UN FICHER

- > Créer le fichier `test.txt` :

test.txt

```
1 3 modules intégrés à Node.JS
2 - os
3 - fs
4 - http
```

- > Créer le fichier `module_fs.js` :

module_fs.js

```
1 const fs = require ('fs');
2
3 fs.readFile('test.txt','utf-8',(err,data) => {
4   if(err) {
5     console.log('Erreur de lecture du fichier ',err);
6     return;
7   }
8   console.log("Données lues dans test.txt :");
9   console.log(data);
10  });
```

- > Paramètres de `fs.readFile()` :

- ▶ Chemin du fichier à lire
- ▶ Encodage du fichier à lire (facultatif, mais recommandé)
- ▶ Fonction callback à appeler une fois le fichier lu

MODULE FS : LIRE UN FICHER

> Résultat :

test.txt

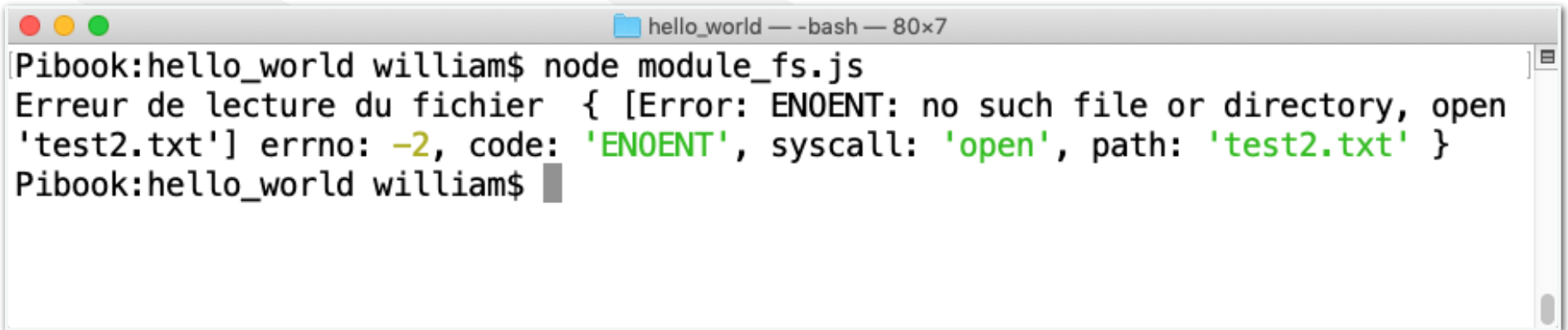
```
1 3 modules intégrés à Node.JS
2 - os
3 - fs
4 - http
```

A terminal window titled 'hello_world — -bash — 80x7' with standard macOS window controls (red, yellow, green buttons). The terminal shows the command 'node module_fs.js' being executed. The output is: 'Données lues dans test.txt :', '3 modules intégrés à Node.JS', '- os', '- fs', '- http'. The prompt 'Pibook:hello_world william\$' is visible at the bottom.

```
Pibook:hello_world william$ node module_fs.js
Données lues dans test.txt :
3 modules intégrés à Node.JS
- os
- fs
- http
Pibook:hello_world william$
```

MODULE FS : LIRE UN FICHER

> En cas d'erreur (mettre `test2.txt` au lieu de `test.txt`) :



```
hello_world — -bash — 80x7
Pibook:hello_world william$ node module_fs.js
Erreur de lecture du fichier { [Error: ENOENT: no such file or directory, open
'test2.txt'] errno: -2, code: 'ENOENT', syscall: 'open', path: 'test2.txt' }
Pibook:hello_world william$
```

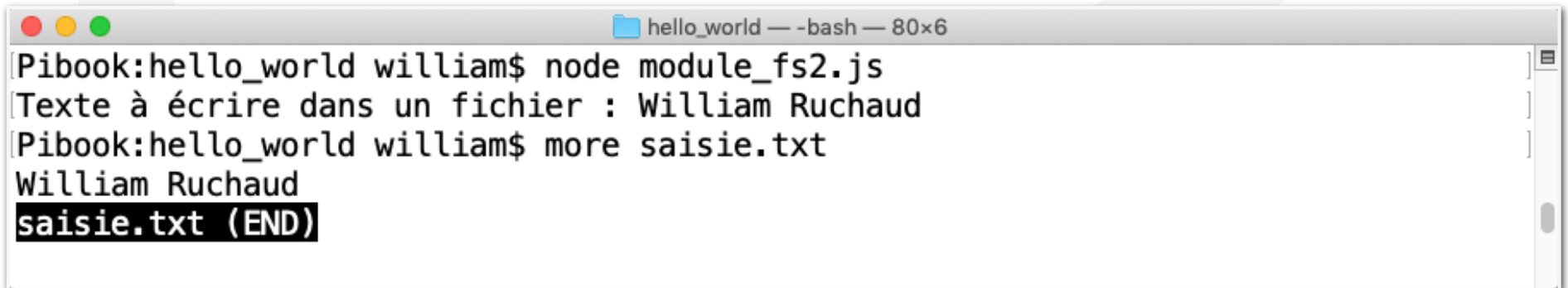
MODULE FS : ÉCRIRE DANS UN FICHER

- > Créer le fichier `module_fs2.js` :

`module_fs2.js`

```
1  const readline = require ('readline');
2  const fs = require ('fs');
3
4  const rl = readline.createInterface({
5    input: process.stdin,
6    output: process.stdout
7  });
8
9  rl.question('Texte à écrire dans un fichier : ',(data)=>{
10    fs.writeFile('saisie.txt',data,'utf-8', (err) =>{
11      if(err) {
12        console.log("Erreur écriture fichier "+err);
13      }
14    });
15    rl.close();
16  });
```

- > Exécuter : `node module_fs2.js`

A terminal window titled 'hello_world — -bash — 80x6' showing the execution of the script. The user enters 'node module_fs2.js', the prompt changes to 'Texte à écrire dans un fichier :', the user enters 'William Ruchaud', the prompt returns to 'Pibook:hello_world william\$', and the user enters 'more saisie.txt'. The output shows 'William Ruchaud' and 'saisie.txt (END)'.

```
Pibook:hello_world william$ node module_fs2.js
Texte à écrire dans un fichier : William Ruchaud
Pibook:hello_world william$ more saisie.txt
William Ruchaud
saisie.txt (END)
```

Node.js

Node.js

Partie 3



MODULE HTTP : SERVEUR WEB

- > Créer le fichier `module_http.js` :

`module_http.js`

```
1  const http = require('http');
2
3  http.createServer((request,response)=>{
4      response.write("Premiere requete HTTP avec Node.JS")
5      response.end();
6  }).listen(8080);
```

- > Détails :

- ▶ Ligne 3 : **`createServer`** : attend un callback avec deux paramètres (noms au choix)
 - **`request`** : contient le détail de la requête (données de formulaire, paramètres de l'URL, méthode...)
 - **`response`** : contenu de la réponse
- ▶ Ligne 4 : écriture d'une donnée dans la réponse
- ▶ Ligne 5 : termine la réponse
- ▶ Ligne 6 : mise en écoute du port 8080

MODULE HTTP : SERVEUR WEB

- > Lancer en ligne de commande : `node module_http.js`



A terminal window titled "hello_world — node module_http.js — 75x5". The prompt is "Pibook:hello_world william\$". The command "node module_http.js" has been entered and executed.

```
Pibook:hello_world william$ node module_http.js
```

- > Avec un navigateur consulter différentes URL en `http://localhost:8080`
- > Toutes répondent :



MODULE HTTP : SERVEUR WEB

- Exemple avec envoi de données d'un formulaire simple en POST
- ▶ Formulaire HTML (utilisation du serveur interne de PHP)
- ▶ Réception des données en Node.js

The image shows a web browser window and a terminal window. The browser window displays a simple HTML form titled "Formulaire POST" with two input fields: "Nom" (containing "Ruchaud") and "Prénom" (containing "William"). An "Envoyer" button is next to the fields. The terminal window shows the command `node module_http.js` being executed, which outputs the received data as a JSON object: `{ nom: 'Ruchaud', prenom: 'William' }`.

Formulaire POST

Nom : Prénom :

Donnees du formulaire : {"nom":"Ruchaud","prenom":"William"}

```
Pibook:hello_world william$ node module_http.js
Donnée envoyées à l'url :/demo/formulaire
{ nom: 'Ruchaud', prenom: 'William' }
```


MODULE HTTP : SERVEUR WEB

- > Créer le fichier `index.php` :

`index.php`

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <meta http-equiv="X-UA-Compatible" content="ie=edge">
7      <title>Document</title>
8  </head>
9  <body>
10     <h1>Formulaire POST</h1>
11     <form action="http://localhost:8080/demo/formulaire" method="POST">
12         <label for="nom">Nom :</label>
13         <input id="nom" name="nom" type="text">
14         <label for="prenom">Prénom :</label>
15         <input id="prenom" name="prenom" type="text">
16         <button>Envoyer</button>
17     </form>
18 </body>
19 </html>
```

- > Noter l'URL de destination du formulaire
- > Lancer en exécutant : `php -S localhost:8090`
- > Consulter avec le navigateur : <http://localhost:8090>

MODULE HTTP : SERVEUR WEB

> Modifier le fichier `module_http.js` :

`module_http.js`

```
1  const http = require('http');
2  const { parse } = require('querystring');
3
4  http.createServer((request, response) => {
5      if (request.method === "POST") {
6          let body = '';
7          request.on('data', (chunk) => {
8              body += chunk.toString();
9          });
10         request.on('end', () => {
11             let data = parse(body);
12             console.log("Donnée envoyées à l'url : " + request.url);
13             console.log(data);
14             response.write("Donnees du formulaire : " + JSON.stringify(data));
15             response.end();
16         });
17     } else {
18         response.write('Reponse HTTP par default');
19         response.end();
20     }
21 }).listen(8080);
```

> Noter :

- ▶ `request.method` : méthode HTTP de la requête
- ▶ `request.url` : URL de la requête

MODULE HTTP : SERVEUR WEB

- > Lancer le programme Node.JS :
`node module_http.js`
- > Dans le navigateur entrer des données
- > Observer :
 - ▶ 8080 = Node.JS, 8090 = PHP
 - ▶ L'affichage dans le navigateur
 - ▶ L'affichage dans la fenêtre de commande

```
Pibook:hello_world william$ node module_http.js
Donnée envoyées à l'url :/demo/formulaire
{ nom: 'Ruchaud', prenom: 'William' }
```

MODULE HTTP : SERVEUR WEB

> Explications :

module_http.js

```
1  const http = require('http');
2  const { parse } = require('querystring');
3
4  http.createServer((request, response) => {
5    if (request.method === "POST") {
6      let body = '';
7      request.on('data', (chunk) => {
8        body += chunk.toString();
9      });
10     request.on('end', () => {
11       let data = parse(body);
12       console.log("Donnée envoyées à l'url : " + request.url);
13       console.log(data);
14       response.write("Donnees du formulaire : " + JSON.stringify(data));
15       response.end();
16     });
17   } else {
18     response.write('Reponse HTTP par default');
19     response.end();
20   }
21 }).listen(8080);
```

Nécessaire pour

- > Lignes 7-9 : réception des données dans un objet Buffer (chunk)
 - ▶ `chunk.toString()` met les données au format URL (nom=ruchaud&...)
- > Ligne 10 : fin de la réception, `parse()` transforme body en objet.

MODULE HTTP : SERVEUR WEB

> Remarque :

- ▶ Le serveur écoute "toutes les URLs", quel que soit le chemin
- ▶ Aspect typique de Node.JS : usage de beaucoup de callbacks

MODULE HTTP : SERVEUR WEB

> Envoyer du HTML :

- ▶ Par défaut la réponse envoyée est "brute"
- ▶ Envoi de balise HTML possible, mais repose sur "le bon comportement des navigateurs"
 - Nécessité d'envoyer des en-têtes indiquant une communication HTML correcte
- ▶ Au minimum : code réponse, type de contenu

> Principe de l'exemple :

- ▶ Répondre 200 pour / 404 pour le reste des URLs
- ▶ Important surtout pour les codes d'erreur
- ▶ Type **text/html**

MODULE HTTP : SERVEUR WEB

> Exemple :

```
module_http.js
1  const http = require('http');
2
3  http.createServer((request,response)=>{
4      switch(request.url) {
5          case '/':
6              response.writeHead(200,{
7                  'Content-type':'text/html'
8              });
9              response.write("Page d'accueil");
10             response.end();
11             break;
12             default:
13                 response.writeHead(404,{
14                     'Content-type':'text/html'
15                 });
16                 response.write('Page not found');
17                 response.end();
18             }
19     }).listen(8080);
```

> `response.writeHead()` permet d'envoyer des en-têtes HTTP.

> Tester en observant le debugger de Chrome avec les URLs :

▶ `/` : code 200

▶ `/quelquechose` : erreur 404

MODULE HTTP : SERVEUR WEB

- > Pour réaliser un serveur Web simpliste façon Apache
 - ▶ Si un fichier html demandé existe : le servir
 - ▶ Sinon : erreur 404
- > Nécessité de gérer l'URL / pour chercher `index.html`
- > Note : ne fonctionne pas avec autre chose que du HTML (pas d'images...)

Node.js

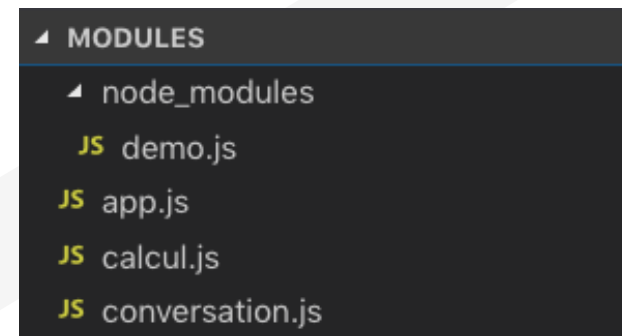
Node.js

Partie 4



CRÉER SES PROPRES MODULES

- > Utilisation du mot-clé **exports** pour indiquer ce qui est "exporté" (utilisable après un require)
- > Plusieurs façons d'écrire possible
- > Comportement de **require** :
 - ▶ Sans indication de chemin : recherche dans le dossier node_modules
 - ▶ Avec indication de chemin : recherche à l'emplacement indiqué
- > L'exemple qui suit est incrémental. Créer un nouveau projet "modules" avec les fichiers suivants :
 - ▶ **demo.js** est le seul fichier dans le dossier **node_modules**
 - ▶ **app.js** sera le fichier à exécuter à chaque fois



CRÉER SES PROPRES MODULES

> Exemple :

app.js

```
1 const conv = require('./conversation');
2
3 conv.direBonjour();
4
5 console.log(conv.auteur);
6 conv.auteur = "coucou";
7 console.log(conv.auteur);
8
9 conv.direAuRevoir();
```

conversation.js

```
1 exports.direBonjour = function(){
2   console.log("Bonjour");
3 }
4
5 exports.auteur = "William Ruchaud";
6
7 direAuRevoir = function() {
8   console.log("Au revoir");
9 }
```

```
modules — -bash — 75x20
...3_nodejs/hello_world — php -S localhost:8090 ...
...ata/3iL/www/b3_nodejs/modules — -bash +
Bonjour
William Ruchaud
coucou
/Volumes/Data/3iL/www/b3_nodejs/modules/app.js:9
conv.direAuRevoir();
  ^

TypeError: conv.direAuRevoir is not a function
    at Object.<anonymous> (/Volumes/Data/3iL/www/b3_nodejs/modules/app.js:9:6)
    at Module._compile (internal/modules/cjs/loader.js:688:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:699:10)
    at Module.load (internal/modules/cjs/loader.js:598:32)
    at tryModuleLoad (internal/modules/cjs/loader.js:537:12)
    at Function.Module._load (internal/modules/cjs/loader.js:529:3)
    at Function.Module.runMain (internal/modules/cjs/loader.js:741:12)
    at startup (internal/bootstrap/node.js:285:19)
    at bootstrapNodeJSCore (internal/bootstrap/node.js:739:3)
Pibook:modules william$
```

Non exporté

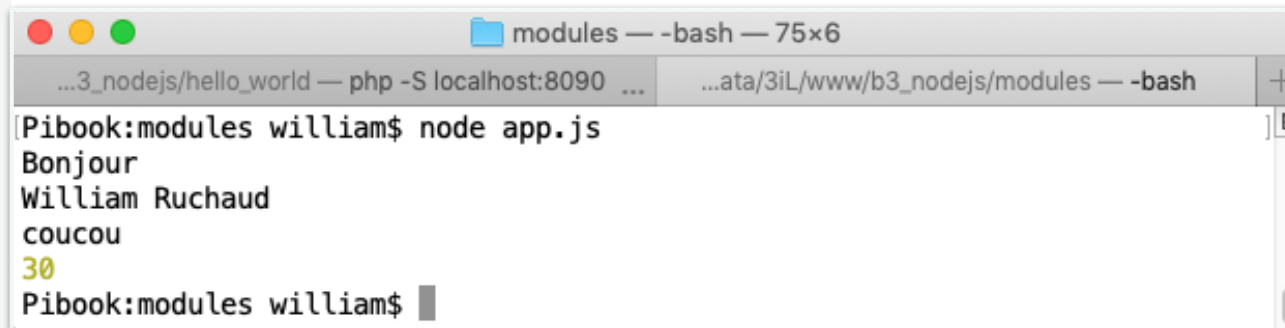
CRÉER SES PROPRES MODULES

> Exemple :

app.js

```
1  const conv = require('./conversation');
2  const { produit } = require('./calcul');
3
4  conv.direBonjour();
5
6  console.log(conv.auteur);
7  conv.auteur = "coucou";
8  console.log(conv.auteur);
9
10 //conv.direAuRevoir();
11
12 console.log(produit(5,6));
13
```

N'importe que la fonction produit()



A terminal window titled 'modules — -bash — 75x6' showing the execution of the app.js file. The prompt is 'Pibook:modules william\$'. The command 'node app.js' has been entered. The output is: 'Bonjour', 'William Ruchaud', 'coucou', and '30' (highlighted in yellow). The prompt is now 'Pibook:modules william\$'.

```
Pibook:modules william$ node app.js
Bonjour
William Ruchaud
coucou
30
Pibook:modules william$
```

CRÉER SES PROPRES MODULES

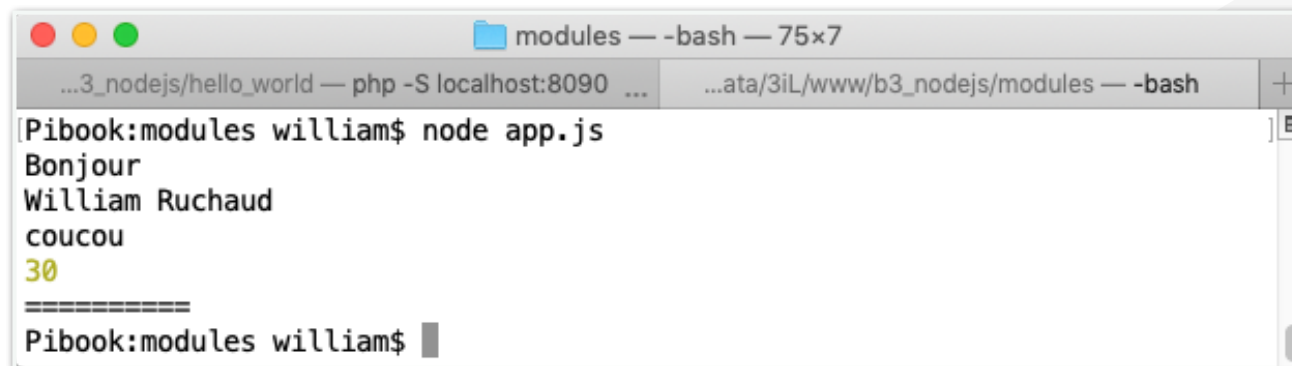
> Exemple :

app.js

```
1  const conv = require('./conversation');
2  const { produit } = require('./calcul');
3  const demo = require('demo');
4
5  conv.direBonjour();
6
7  console.log(conv.auteur);
8  conv.auteur = "coucou";
9  console.log(conv.auteur);
10
11 //conv.direAuRevoir();
12
13 console.log(produit(5,6));
14
15 demo.trait();
```

node_modules/demo.js

```
1  exports.trait = function() {
2    console.log('=====');
3  }
```



The terminal window shows the execution of the app.js file. The output is as follows:

```
Pibook:modules william$ node app.js
Bonjour
William Ruchaud
coucou
30
=====
Pibook:modules william$
```