

Serveur Http



EXPLORER

OPEN EDITORS

Welcome

JS HelloWorld.js

FIRSTAPP

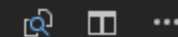
JS HelloWorld.js



OUTLINE

Welcome

JS HelloWorld.js x



```
1 var http = require('http');
2 
3 var server = http.createServer(function(req, res) {
4   res.writeHead(200);
5   res.end('Hello world !');
6 });
7 server.listen(8080);
```

PROBLEMS

TERMINAL

...

1: node



Windows PowerShell

Copyright (C) Microsoft Corporation. Tous droits réservés.

PS C:\Users\user\Desktop\FirstApp> node HelloWorld.js

```
1 var http = require('http');
```

require effectue un appel à une bibliothèque de Node.js, ici la bibliothèque "http" qui nous permet de créer un serveur web. Il existe des tonnes de bibliothèques comme celle-là, la plupart pouvant être téléchargées avec NPM, le gestionnaire de paquets de Node.js (on apprendra à l'utiliser plus tard).

La variable http représente un objet JavaScript qui va nous permettre de lancer un serveur web. C'est justement ce qu'on fait avec :

```
1 var server = http.createServer();
```

On appelle la fonction `createServer()` contenue dans l'objet `http` et on enregistre ce serveur dans la variable `server`. Vous remarquerez que la fonction `createServer` prend un paramètre... et que ce paramètre est une fonction ! C'est pour ça que l'instruction est un peu compliquée, puisqu'elle s'étend sur plusieurs lignes :

```
1 var server = http.createServer(function(req, res) {  
2   res.writeHead(200);  
3   res.end('Salut tout le monde !');  
4 });
```

Tout le code ci-dessus correspond à l'appel à `createServer()`. Il comprend en paramètre **la fonction à exécuter quand un visiteur se connecte à notre site.**

Notez que vous pouvez faire ça en deux temps comme je vous l'avait dit. La fonction à exécuter est la fonction de *callback*.

```
1 res.writeHead(200);  
2 res.end('Salut tout le monde !');
```

On renvoie le code 200 dans l'en-tête de la réponse, qui signifie au navigateur "OK tout va bien" (on aurait par exemple répondu 404 si la page demandée n'existait pas). Il faut savoir qu'en plus du code HTML, le serveur renvoie en général tout un tas de paramètres en en-tête. Il faut connaître la norme HTTP qui indique comment clients et serveurs doivent communiquer pour bien l'utiliser. Voilà encore un exemple de la complexité due au fait que Node.js est bas niveau... Mais en même temps ça nous fait comprendre tout un tas de choses. 😊

Ensuite, on termine la réponse (avec `end()`) en envoyant le message de notre choix au navigateur. Ici, on n'envoie même pas de HTML, juste du texte brut.

Enfin, le serveur est lancé et "écoute" sur le port 8080 avec l'instruction :

javascript

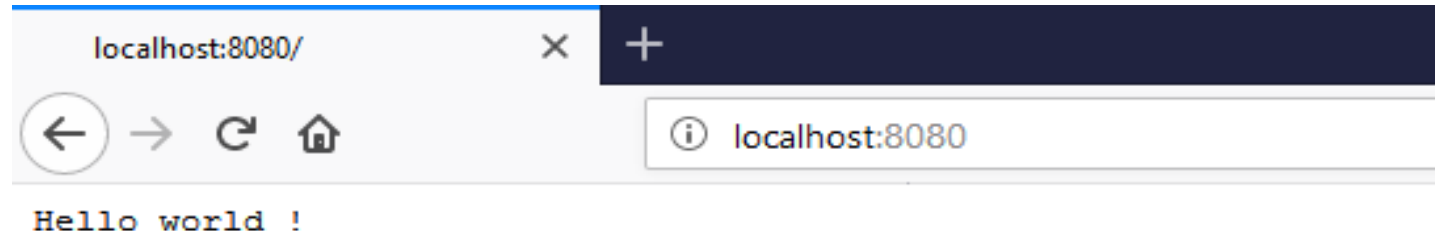
```
1 server.listen(8080);
```

Tester le serveur HTTP

- Pour tester votre premier serveur, rendez-vous dans la console et tapez :

```
node HelloWorld.js
```

- La console n'affiche rien, c'est tout à fait normal. Ouvrez maintenant votre navigateur et rendez-vous à l'adresse `http://localhost:8080`. Vous allez vous connecter sur votre propre machine sur le port 8080 sur lequel votre programme Node.js est en train d'écouter !



Vous vous souvenez comment on écrit dans l'en-tête de la réponse avec Node.js ? Nous avons écrit ceci :

javascript

```
1 res.writeHead(200);
```

Nous avons seulement indiqué le code de réponse 200 qui signifie "OK, pas d'erreur". Nous devons rajouter un paramètre qui indique le type MIME de la réponse. Pour HTML, ce sera donc :

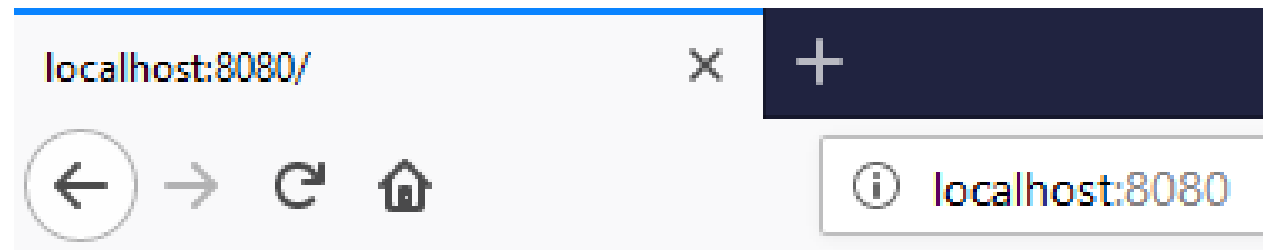
javascript

```
1 res.writeHead(200, {"Content-Type": "text/html"});
```

- Du texte brut : text/plain
- Du HTML : text/html
- Du CSS : text/css
- Une image JPEG : image/jpeg
- Une vidéo MPEG4 : video/mp4
- Un fichier ZIP : application/zip
- etc.

Améliorons un peu notre serveur

```
1 var http = require('http');
2
3 var server = http.createServer(function(req, res) {
4   res.writeHead(200, {"Content-Type": "text/html"});
5   res.end('<p>Hello <strong>College Rosemont</strong> !</p>');
6 });
7 server.listen(8080);
```



Hello College Rosemont !

Déterminer la page appelée et les paramètres

Comment récupérer :

- ▶ Le nom de la page demandée (/mapage, /page.html, /dossier/autrepage...)
- ▶ Les paramètres qui circulent dans l'URL (ex : `http://localhost:8080/mapage?nom=dupont&prenom=robert`).

Quelle est la page demandée par le visiteur ?

Pour récupérer la page demandée par le visiteur, on va faire appel à un nouveau module de Node appelé "url". On demande son inclusion avec :

javascript

```
1 var url = require("url");
```

Ensuite, il nous suffit de "parser" la requête du visiteur comme ceci pour obtenir le nom de la page demandée :

javascript

```
1 url.parse(req.url).pathname;
```

```
1  var http = require('http');
2  var url = require('url');
3
4  var server = http.createServer(function(req, res) {
5      var page = url.parse(req.url).pathname;
6      console.log(page);
7      res.writeHead(200, {"Content-Type": "text/plain"});
8      if (page == '/') {
9          res.write('Vous etes a l\'accueil');
10     }
11     else if (page == '/Niveau1') {
12         res.write('Vous etes au niveau 1 !');
13     }
14     else if (page == '/Niveau2/Niveau22') {
15         res.write('Vous etes au niveau 22 !');
16     }
17     res.end();
18 });
19 server.listen(8080);
```

Quels sont les paramètres ?

```
1 url.parse(req.url).query
```

Le problème, c'est qu'on vous renvoie toute la chaîne sans découper au préalable les différents paramètres. Heureusement, il existe un module Node.js qui s'en charge pour nous : `querystring` !

Incluez ce module :

javascript

```
1 var querystring = require('querystring');
```

Vous pourrez ensuite faire :

javascript

```
1 var params = querystring.parse(url.parse(req.url).query);
```

```
1 var http = require('http');
2 var url = require('url');
3 var querystring = require('querystring');
4
5 var server = http.createServer(function(req, res) {
6     var params = querystring.parse(url.parse(req.url).query);
7     res.writeHead(200, {"Content-Type": "text/plain"});
8     if ('prenom' in params && 'nom' in params) {
9         res.write('Vous vous appelez ' + params['prenom'] + ' ' + params['nom']);
10    }
11    else {
12        res.write('Vous devez bien avoir un prénom et un nom, non ?');
13    }
14    res.end();
15 });
16 server.listen(8080);
```

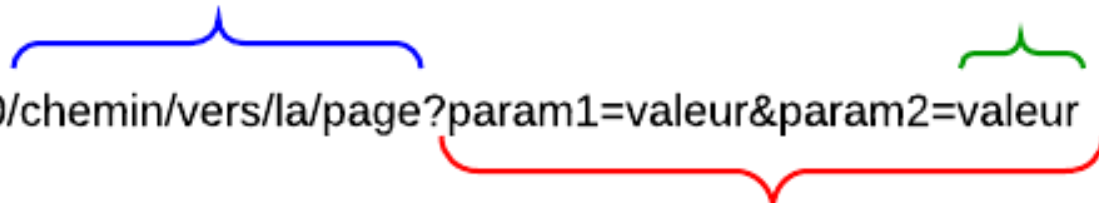
Essayez d'aller sur <http://localhost:8080?prenom=Robert&nom=Dupont> pour voir, puis changez le prénom et le nom pour les remplacer par les vôtres !

Schéma résumé

```
querystring.parse(url.parse(req.url).query)['param2']
```

```
url.parse(req.url).pathname
```

`http://localhost:8080/chemin/vers/la/page?param1=valeur¶m2=valeur`



```
url.parse(req.url).query
```


Références:

- ▶ <https://openclassrooms.com/fr/courses/1056721-des-applications-ultra-rapides-avec-node-js>