

Atelier :**Création d'une application avec Node.js****Objectif :**

Cet atelier consiste à construire un serveur web avec Node.js, à retourner du code HTML et de déterminer la page appelée et ses paramètres.

Construction d'un serveur web:

1. Créez un dossier sous le nom de **Node.js**
2. Dans ce dossier, créez un fichier **server.js** comme suit:

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

Figure 1: server.js

Il crée un mini-serveur web qui renvoie un message "Salut les webistes", quelle que soit la page demandée. Ce serveur est lancé sur le port 8080 à la dernière ligne.

Décomposons le code :

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

require effectue un appel à une bibliothèque de Node.js "**http**" qui nous permet de créer un serveur web.

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**.

La fonction **createServer()** prend un paramètre qui est une fonction:

```
var server = http.createServer(function(req, res) {  
    res.writeHead(200);  
    res.end('Salut les webistes !');  
});
```

La fonction en paramètre est exécutée quand un utilisateur se connecte à votre site.

⇒ C'est une fonction de **callback**

La fonction de callback est donc appelée à chaque fois qu'un visiteur se connecte à notre site. Elle prend 2 paramètres :

- La requête du visiteur (**req**) : cet objet contient toutes les informations sur ce que le visiteur a demandé. On y trouve le nom de la page appelée, les paramètres, les éventuels champs de formulaires remplis...
- La réponse que vous devez renvoyer (**res**) : c'est cet objet qu'il faut remplir pour donner un retour au visiteur. Au final, **res** contiendra en général le code HTML de la page à renvoyer au visiteur.

```
res.writeHead(200);  
res.end('Salut les webistes !');
```

On renvoie le code **200** dans l'en-tête de la réponse, qui signifie au navigateur "OK , la page est retrouvée" (on aurait par exemple répondu 404 si la page demandée n'existait pas).

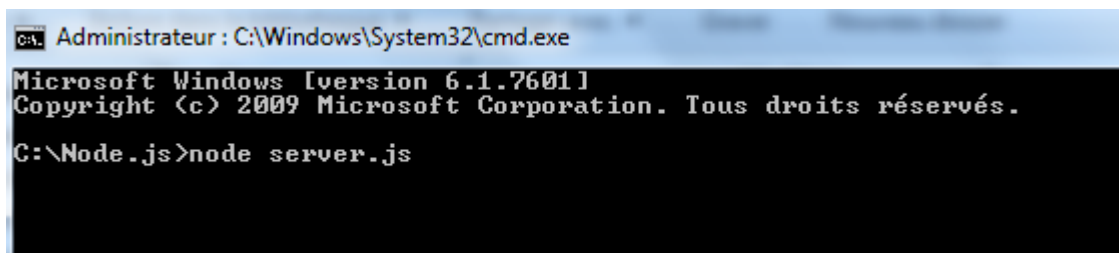
Ensuite, on termine la réponse (avec **end()**) en envoyant le message de notre choix au navigateur.

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

Enfin, le serveur est lancé et "**écoute**" sur le port **8080**.

3. Testez le premier serveur crée avec Node.js, ouvrez une console dans le dossier où se trouve votre fichier **server.js** et entrez la commande :

node server.js



```
Administrateur : C:\Windows\System32\cmd.exe
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tous droits réservés.
C:\Node.js>node server.js
```

4. Rendez-vous sur <http://localhost:8080/>



Retourner du code HTML :

En respectant la **norme http** inventée par **Tim Berners-Lee**, le serveur doit indiquer le type de données qu'il s'apprête à envoyer au client :

- 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.

Ce sont ce qu'on appelle les types MIME. Ils sont envoyés dans l'en-tête de la réponse du serveur.

5. Ajoutez un paramètre qui indique le type MIME de la réponse. Pour HTML, ce sera donc :

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

6. Ajoutez le code HTML suivant à la réponse:

```
res.end('<p>Voici un paragraphe <strong>HTML</strong> !</p>');
```

Le code final sera comme suit :

```
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>Voici un paragraphe <strong>HTML</strong> !</p>');
6  });
7  server.listen(8080);
```

7. Redémarrez votre serveur en relançant la commande **node server.js** dans la console
8. Rendez-vous sur <http://localhost:8080/>



Voici un paragraphe **HTML** !

- Utilisez la commande **res.write()** au lieu de **res.end()** pour mieux découper et découpler le code.

res.end() doit toujours être appelé en dernier pour terminer la réponse et faire en sorte que le serveur envoie le résultat au client.

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

var server = http.createServer(function(req, res) {
  res.writeHead(200, {"Content-Type": "text/html"});
  res.write('<!DOCTYPE html>' +
    '<html>' +
    '  <head>' +
    '    <meta charset="utf-8" />' +
    '    <title>Ma page Node.js !</title>' +
    '  </head>' +
    '  <body>' +
    '    <p>Voici un paragraphe <strong>HTML</strong> !</p>' +
    '  </body>' +
    '</html>');
  res.end();
});

server.listen(8080);
```

Déterminer la page appelée et les paramètres :

Nous allons découvrir comment récupérer :

- Le nom de la page demandée (/home, /page.html, /dossier/autrepag...)
- Les paramètres qui circulent dans l'URL (ex : <http://localhost:8080/mapage?id=3008&login=web>).

- 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 :

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

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

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

12. Le code final devient comme suit :

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

var server = http.createServer(function(req, res) {
  var page = url.parse(req.url).pathname;
  console.log(page);
  res.writeHead(200, {"Content-Type": "text/plain"});
  res.write('Bien on avance!');
  res.end();
});
server.listen(8080);
```

13. En exécutant ce code on aura le même message affiché 'Bien on avance' dans le navigateur quelque soit la page appelée !
14. Ajoutez des conditions pour gérer les URLS demandées par le visiteur :

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

var server = http.createServer(function(req, res) {
  var page = url.parse(req.url).pathname;
  console.log(page);
  res.writeHead(200, {"Content-Type": "text/plain"});
  if (page == '/') {
    res.write('Vous êtes dans la page d\'accueil');
  }
  else if (page == '/Contact') {
    res.write('Vous êtes dans la page Contact !');
  }
  else if (page == '/Affichage/l/user') {
    res.write('Affichez l\'utilisateur qui a l\'id 1 !');
  }

  res.end();
});
server.listen(8080);
```

15. Ajoutez le code nécessaire pour afficher un message d'erreur « 404 not found ! » si le visiteur demande une page inconnue.
16. Pour récupérer des paramètres envoyés dans l'URL, il suffit de faire appel à :

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

17. Pour découper la chaîne des paramètres, on utilise le module suivant :

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

18. Ensuite, on récupère la liste des paramètres dans une variable :

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

19. La variable "**params**" représente un tableau de paramètres. Pour récupérer le paramètre "id" par exemple, il suffira d'écrire : **params['id']**.
20. Exécutez le code complet qui affiche l'id et le login d'un utilisateur :

```
var http = require('http');
var url = require('url');
var querystring = require('querystring');

var server = http.createServer(function(req, res) {
  var params = querystring.parse(url.parse(req.url).query);
  res.writeHead(200, {"Content-Type": "text/plain"});
  if ('id' in params && 'login' in params) {
    res.write('Votre id est ' + params['id'] +
      ' et votre login' + params['login']);
  }
  else {
    res.write('Veuillez saisir votre id et login!');
  }
  res.end();
});
server.listen(8080);
```

21. Combinez ce code et le précédent pour gérer à la fois la page et les paramètres.