WebServer

NodeJS

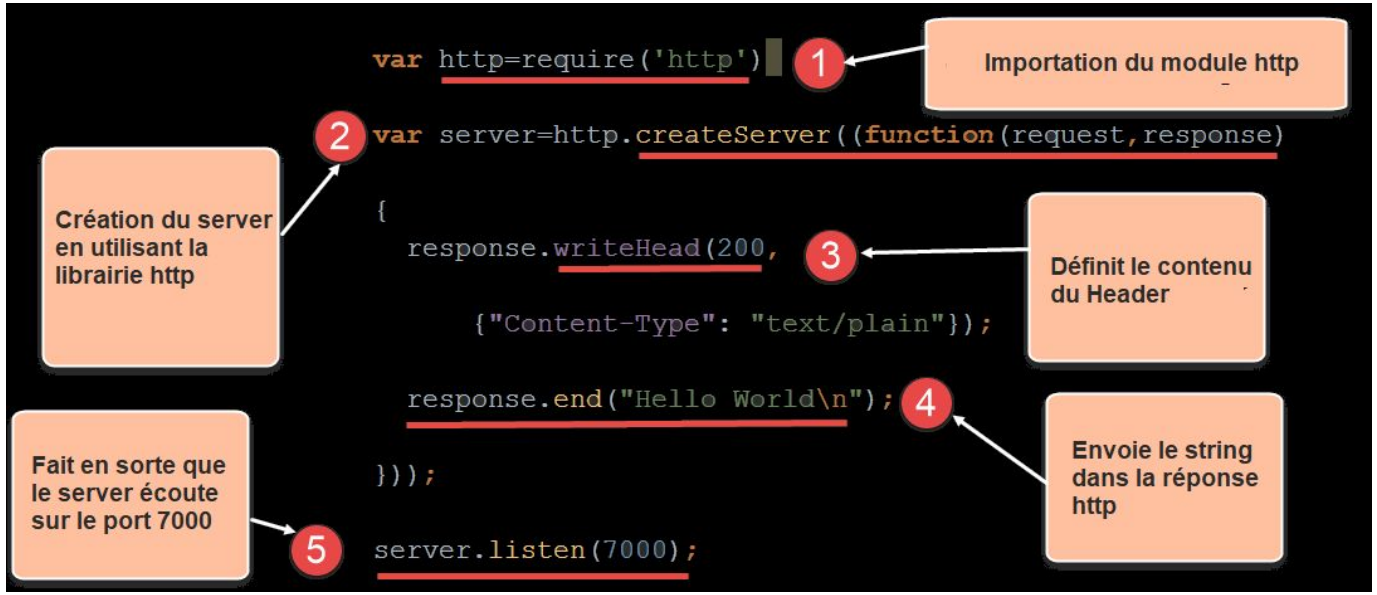
WebServer

NodeJS est généralement utilisé pour créer des applications Clients/Server.

Il existe plusieurs modules tels *http* ou *request* qui peuvent nous aider à mettre en place un WebServer ou a effectuer une requête http

Utilisation du module *http*

Voici un simple exemple de mise en place d'un serveur Web



WebServer

Digression pratique

Lorsque nous changeons notre code, nous devons stopper et relancer la commande node…

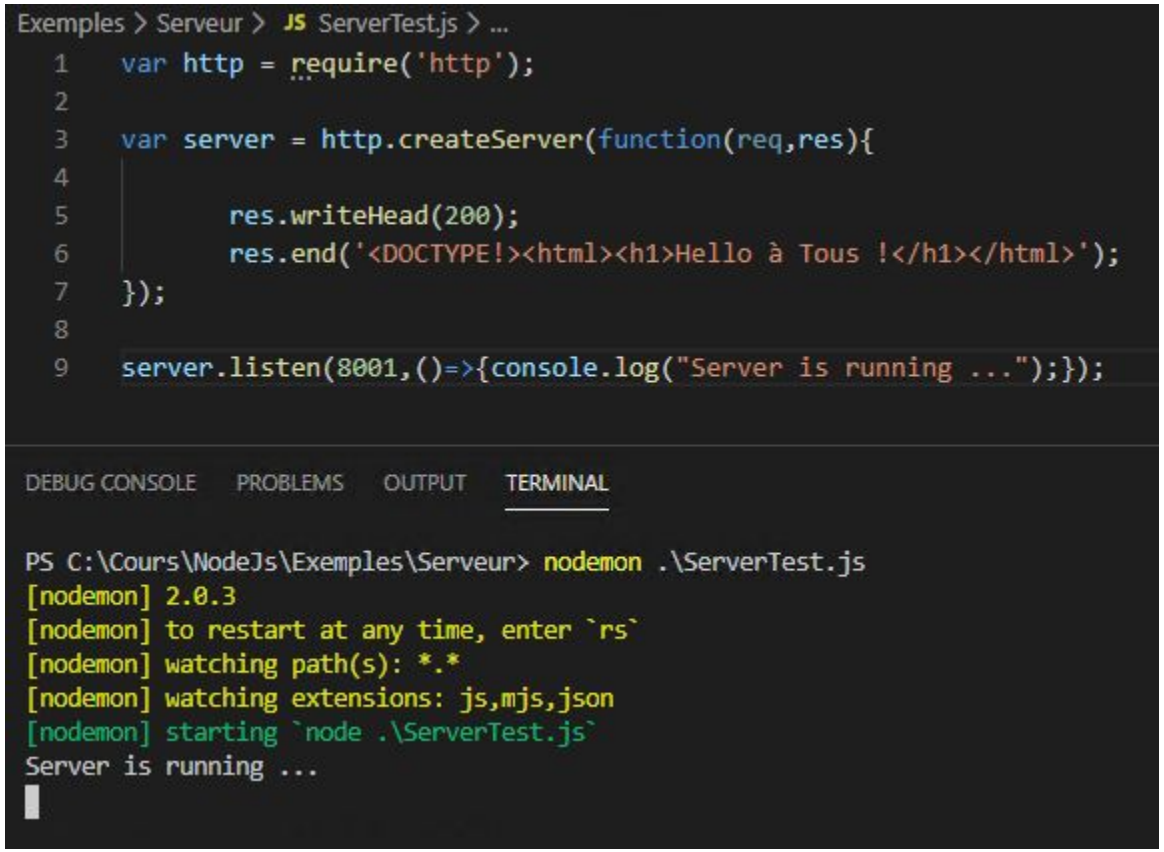
Cela peut s'avérer « énervant ».

Il est possible d'utiliser le package *nodemon* qui est un simple wrapper permettant de relancer pour nous nodejs lorsqu'un changement est détecté sur un fichier

Pour l'utiliser :

1- npm install -g nodemon

2- on remplace node par nodemon pour lancer notre app.



WebServer

Request

le premier argument de la fonction associée au *createServer* est l'http Request qui est de type *IncommingMessage* sous NodeJS.

Cet Objet *IncommingMessage* permet d'accéder aux différentes propriété de la requête tel le statuscode, le headers,…

Il est intéressant de voir que lorsqu'on demande l'affichage d'une page simple, le fonction callback du *http.CreateServer* est appelé plus d'une fois.

Cela s'explique par la demande du Navigateur.

Exemples > Serveur > IncommingMessageSample > JS ServerTest.js > server > http.createServer() callback

```
1   var http = require('http');
2   let cpt =1;
3   var server = http.createServer(function(req,res){
4           console.log(cpt +" - " +req.url);
5           res.writeHead(200);
6
7           res.end('<DOCTYPE!><html><h1>Hello à Tous !</h1></html>');
8           cpt++;
9   });
10
11  server.listen(8001,()=>{console.log("Server is running ... http://127.0.0.1:8001");});
```

DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

```
PS C:\Cours\NodeJs\Exemples\Serveur\IncommingMessageSample> nodemon .\ServerTest.js
[nodemon] 2.0.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node .\ServerTest.js`
Server is running ... http://127.0.0.1:8001
1 - /
2 - /favicon.ico
```

WebServer

GET

Si nous désirons récupérer les paramètres se trouvant dans l'url, nous utiliserons *req.url* comme dans l'exemple précédent.

Cependant, pour nous faciliter la vie, nous utiliserons un module supplémentaire appelé ***url*** ***(***https://nodejs.org/api/url.html)

Grâce à cette librairie, nous pourrons facilement « parser » notre querystring et obtenir les paramètres transmis

Exemples > Serveur > QueryStringSample > JS ServerTest.js > server > http.createServer() callback

```
1   var http = require('http');
2   var url = require('url');
3   let cpt =1;
4   var server = http.createServer(function(req,res){
5           if(cpt==1)
6           {
7                   console.log(url.parse(req.url,true).query);
8                   console.log("Host : " + url.parse(req.url,true).host);
9                   console.log("Pathname : "+ url.parse(req.url,true).pathname);
10                  console.log("search : " + url.parse(req.url,true).search );
11          }
12
13          res.writeHead(200);
14
15          res.end('<DOCTYPE!><html><h1>Hello à Tous !</h1></html>');
16          cpt++;
17  });
18
19  server.listen(8001,()=>{console.log("Server is running ... http://127.0.0.1:8001");});
```

DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

```
PS C:\Cours\NodeJs\Exemples\Serveur\QueryStringSample> nodemon .\ServerTest.js
[nodemon] 2.0.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node .\ServerTest.js`
Server is running ... http://127.0.0.1:8001
[Object: null prototype] { id: '2', titre: 'Bonjour' }
Host : null
Pathname : /app.js
search : ?id=2&titre=Bonjour
```

WebServer

POST

En premier lieu, nous devons vérifier si nous sommes bien face à une requête POST.

Ensuite nous devons gérer deux événements :

- data on reçoit les données du formulaire
- end les données ont été transférées

Enfin, nous devrons lire un *ReadableStream* qui contient les informations du POST et pour cela, nous utilisons la méthode *parse* de la librairie *querystring*.

```
1  const  http= require('http');
2  const  { parse }  = require('querystring');
3  const requestListener = function(req,res)
4  {
5      if (req.method === 'POST')
6      {
7          let body = '';
8          req.on('data', form => { body += form.toString(); }); //On récupère le coprs de la requête
9          req.on('end', () => {   //Tout le stream a été transmis
10             console.log(parse(body)); //on utilise la méthode parse de la librairie querystring
11             res.writeHead(204);
12             res.end();
13         });
14     }
15     else
16     {
17     let body = `<!doctype>
18                 <html><body>
19                     <form action="/" method="post">
20                         <input type="text" name="fname" /><br />
21                         <input type="number" name="age" /><br />
22                         <input type="file" name="photo" /><br />
23                         <button>Save</button>
24                     </form>
25                 </body></html>`;
26     res.writeHead(200,{
27         'Content-Length': Buffer.byteLength(body),
28         'Content-Type': 'text/html'
29       });
30     res.end(body);
31     }
32 }
33 var server = http.createServer();
34 server.on('request',requestListener);
35 server.listen(8001,()=>{console.log("Server is running on http://localhost:8001/");});
```

```
DEBUG CONSOLE   PROBLEMS   OUTPUT   TERMINAL                    1: node

Server is running on http://localhost:8001/
[Object: null prototype] {
  fname: 'Mike',
  age: '5',
  photo: 'package.json'
}
```

WebServer

Tips :

- Si nous désirons envoyer de l'html plutôt que du simple texte à notre navigateur, nous devons définir le *content-type* dans le header de la response.

```
res.writeHead(200,{
    'Content-Length': Buffer.byteLength(body),
    'Content-Type': 'text/html'
  });
```

- Si nous voulons écrire un string sur plusieurs lignes (pour une question de lisibilité), écrivez

`<h1>…</h1>`

plutôt que

'<h1>…</h1>'