La plateforme NodeJS

Plan

NodeJS

JavaScript : rôle

permet d'écrire des scripts qui seront exécutés coté client (le navigateur du visiteur exécute le script JS et effectue des actions (modifications, animations...) sur la page web)

NodeJS : rôle

permet d'écrire du code JavaScript qui s'exécute coté serveur (comme le fait, PHP, Java EE...)

NodeJS

NodeJS : caractéristiques

- est un projet open source
- développé en C++
- basé sur l'interpréteur du JavaScript
- utilise le moteur d'exécution V8 de Google Chrome
- est event-driven et utilise seulement des non-blocking I/O API (et des callbacks)

NodeJS

NodeJS : caractéristiques

- est un projet open source
- développé en C++
- basé sur l'interpréteur du JavaScript
- utilise le moteur d'exécution V8 de Google Chrome
- est event-driven et utilise seulement des non-blocking I/O API (et des callbacks)

Dans le monde

Il est utilisé par Groupon, IBM, LinkedIn, Microsoft, Paypal, Yahoo...

NodeJS

V8 : caractéristiques

- est un projet open source aussi
- créé par Google en 2008
- codé en C++

NodeJS

V8 : caractéristiques

- est un projet open source aussi
- créé par Google en 2008
- codé en C++

V8 : rôle

- compile le code JS en code machine et l'exécute
- optimise et ré-optimise le code

NodeJS

V8 : caractéristiques

- est un projet open source aussi
- créé par Google en 2008
- codé en C++

V8 : rôle

- compile le code JS en code machine et l'exécute
- optimise et ré-optimise le code

Dans le monde

Il est utilisé par MongoDB, NodeJS

NodeJS

Installation facile

- Aller`https://nodejs.org/en/download/`
- Télécharger la version qui correspond à votre OS
- Installer le fichier téléchargé

NodeJS

Étapes

Ouvrir une console **Cmder** et exécuter les commandes suivantes dans l'ordre

- `mkdir testNodeJS`
- `cd testNodeJS`
- `echo console.log('Hello world'); > test.js`
- `cat test.js`
- `node test.js`

NodeJS

Étapes

Ouvrir une console **Cmder** et exécuter les commandes suivantes dans l'ordre

- `mkdir testNodeJS`
- `cd testNodeJS`
- `echo console.log('Hello world'); > test.js`
- `cat test.js`
- `node test.js`

Et voici le premier Hello world

NodeJS

Exercice

- Considérons les données suivantes :

```
var tab = [1,3,6,8,9];
var element = 5;
```

- Ecrire une fonction qui permet de vérifier si `element` appartient à tab

NodeJS

Solution avec une fonction callback

```
var searchElement = function(data, callback){
  for (var i = 0; i< data.tableau.length; i++)
    if(data.tableau[i]==data.filtre)
      return callback(null,i);
  return callback('Element ' + data.filtre +  ' non
   retrouve dans tableau');
};
var tab = [1,3,6,8,9];
var element = 5;
var data = {tableau : tab, filtre : element} ;
searchElement(data, function (err,result){
  if (err)
    console.error("erreur :" + err)
  else
    console.log(element + " existe a la position "  +
     result)
});
```

NodeJS

Conventions

- Les fonctions callback prennent deux paramètres :
  - un paramètre `err` qui reste vide si la fonction a bien été exécutée, sinon il contient le contenu du message d'erreur
  - un paramètre `result` qui contient le résultat si la fonction n'a pas détecter d'erreurs

NodeJS

Exercice 1

En utilisant les fonctions callback, écrire une fonction qui permet de déterminer le nombre d'occurrence d'une sous-chaîne de caractère ch dans une chaîne de caractère str.

- $ch = ab$
- $str = abbbaaaabaaabb$
- la fonction retourne 3.

NodeJS

Exercice 2

En utilisant les fonctions callback, écrire une fonction qui permet de déterminer le nombre d'occurrence des éléments d'un ensemble de sous-chaîne de caractère tab apparaît dans une chaîne de caractère str

- $tab = [ab, ba, abba]$
- $str = abbbaaaabaaabb$
- les éléments $'ab'$ et $'ba'$ sont dans str, donc notre programme retourne 5.

NodeJS

Trois types de modules

- Des modules qui sont définis dans le noyau du nodeJS : pour les utiliser, il faut juste `require('nomModule');`
- Des modules de la communauté NodeJS : pour les utiliser, il faut les télécharger puis les utiliser via la console
- Nos propres modules : pour les utiliser, il faut les exporter puis les importer avec `require('./nomModule');`

NodeJS

Les modules de la communauté

- sont gérés par NPM (`Node Package Manager`), voir `www.npmjs.com`
- installé automatiquement avec `nodeJS`

NodeJS

Les commandes possibles avec npm

- Installer localement un module `npm install nomModule` (ajoute généralement un répertoire `node_modules` dans le répertoire courant)
- Installer globalement un module `npm install -g nomModule`
- Avoir de l'aide sur une commande `npm help nomCommande`
- Lister les modules installés `npm ls`
- Lister les modules globaux installés `npm ls -g`
- Localiser le node_modules `npm root`
- Initialiser un projet `npm init` (génère un fichier `package.json`)

NodeJS

Les commandes possibles avec npm

- Chercher un module `npm search nomModule`
- Désinstaller un module `npm uninstall nomModule`
- Désinstaller un module global `npm uninstall -g nomModule`
- Linker un module global à un projet local `npm link nomModule`
- Lister les modules qui ne sont pas à jour `npm outdated`
- Mettre à jour les modules `npm update`
- Reconfigurer NPM `npm config`

NodeJS

Exemple

- Dans un nouveau répertoire, initialiser le projet avec `npm init` (un fichier `package.json` sera créé), vérifier son contenu.
- Avec la console, exécuter `npm install --save lodash` ou le raccourci `npm i -S lodash`
- Vérifier l'ajout de lodash dans la section `dependencies` de `package.json` et la création d'un répertoire `node_modules` contenant `lodash`
- Tester le code suivant

```
var math = require('lodash');
console.log(math.map([1,5,3], function (a) {
 return a * 2}));
```

NodeJS

Exporter nos modules

- Créer deux fichiers : `mesModules.js` et `test.js`
- Utiliser la fonction `require` dans `test.js` pour importer les modules définis dans `mesModule.js`

NodeJS

Exporter nos modules

- Créer deux fichiers : `mesModules.js` et `test.js`
- Utiliser la fonction `require` dans `test.js` pour importer les modules définis dans `mesModule.js`

On peut exporter

- des fonctions
- des objets
- des constantes
- des variables
- ...

NodeJS

Dans mesModules.js

```
var direBonjour = function(){
    console.log('Bonjour');
}
module.exports = direBonjour;
```

Dans test.js

```
var  mod = require('./mesModules');
mod();
```

NodeJS

Dans mesModules.js

```
exports.direBonjour = function(){
    console.log('Bonjour');
}
```

Dans test.js

```
var  mod = require('./mesModules');
mod.direBonjour();
```

NodeJS

Dans mesModules.js

```
var direBonjour = function(){
    console.log('Bonjour');
}

var direBonsoir = function(){
    console.log('Bonsoir');
}

module.exports = {sayHello : direBonjour};
```

Dans test.js

```
var  mod = require('./mesModules');
mod.sayHello();
```

On n'est pas obligé de tout exporter

NodeJS

Étapes

- Créer un utilisateur NPM (`npm addUser`)
- Créer un répertoire contenant le(s) fichier(s) de notre module
- Transformer ce répertoire en répository git (`git init`) et associer ce module à un repository Github (`git remote add origin url_repository_github`, `git fetch`, `git add .` , `git commit -m "init"`, `git rebase origin/master` et `git push origin HEAD:master`)
- Initialiser votre projet `npm init`
- Ajouter le `package.json` sur github : `git add .` , `git commit -m "add package.json"` et `git push origin HEAD:master`

NodeJS

Étapes

- Publier sur npm (`npm publish`)
- Aller vérifier sur `npmjs.com`

NodeJS

- Créer trois fichiers : `sync.js`, `async.js` et `salutation.txt`
- Utiliser une bibliothèque existante : fs (file system) (voir la liste complète `https://nodejs.org/api/documentation.html`)

Dans salutation.txt

```
bonjour
bonsoir
salut
aurevoir
```

NodeJS

Dans sync.js

```
var fs= require('fs');
var content = fs.readFileSync('./salutation.txt');
console.log(content.toString());
console.log('end of file');
```

Dans async.js

```
var fs = require('fs');
var content = fs.readFile('./salutation.txt',
 function (err,result){
    if (err)
        return console.error(err);
    return console.log(result.toString());
});
console.log('end of file');
```

NodeJS

- Exécuter les deux fichiers séparément
- Quelle différence ?

NodeJS

Exercice :

- Ecrire un programme NodeJS qui permet de créer un répertoire `monDossier` et trois fichiers `file1.txt`, `file2.txt` et `file3.txt` qui seront situés dans `monDossier`
- Utiliser à la fois des fonctions synchrones et des fonctions asynchrones

NodeJS

Solution

```
var fs = require("fs");
var file = ["file1.txt", "file2.txt", "file3.txt"];

if (fs.existsSync('monDossier'))
  console.error('dossier existe deja');
else
  fs.mkdirSync('monDossier');
for (let i = 0; i < file.length; i++) {
  fs.writeFile('monDossier/' + file[i], 'contenu
   fichier', (err) => {
    if (err)
      console.error(err);
  });
}
```

NodeJS

Exercice :

Ecrire un programme NodeJS qui permet de générer un rapport illustrant le contenu d'un répertoire appelé `monDossier`

NodeJS

Solution

```
let fs = require('fs');
let fichier = "rapportMonDossier.txt";
let dossier = 'monDossier';
let arbo = fs.readdirSync(dossier);
for(i=0; i<arbo.length; i++){
  let data = arbo[i]+"\n";
  fs.appendFileSync(fichier, data);
  if (fs.lstatSync(dossier+"/"+arbo[i]).isDirectory
   ()) {
    let arbo2 = fs.readdirSync(dossier+"/"+arbo[i]);
    for (j=0; j<arbo2.length; j++){
      let data2 = "-"+arbo2[j]+"\n";
      fs.appendFileSync(fichier, data2);
    }
  }
}
```

NodeJS

Exercice :

Ecrire un programme NodeJS qui permet de générer à partir d'un rapport l'arborescence correspondante

NodeJS

Solution

```
let fs = require('fs');
let readline = require('readline');
let fichier = "rapportMonDossier.txt";
let tabLine = [];
let dossier = "monDossier";

const rl = readline.createInterface({
    input: fs.createReadStream(fichier),
    crlfDelay: Infinity
  });

  rl.on('line', (line) => {
    tabLine.push(line);
  });
  rl.on('close', (line) => {
    for (let i=0; i<tabLine.length; i++){
        console.log(tabLine[i+1]);
        if(tabLine[i+1]){
            if (tabLine[i+1][0]!= "-" && tabLine[i][0]!= "-") {
```

Suite

```
                fs.writeFileSync(dossier+"/"+tabLine[i]);
            } else {
                if(tabLine[i][0]!="-"){
                    fs.mkdirSync(dossier+"/"+tabLine[i]);
                    let j=1;
                    while (tabLine[i+j][0]=="-"){
                        ligne = tabLine[i+j].substr(1,
                         tabLine[i+j].length-1);
                        fs.writeFileSync(dossier+"/"+
                         tabLine[i]+"/"+ligne);
                        j++;
                    }
                }
            }
        } else {
            fs.writeFileSync(dossier+"/"+tabLine[i]);
        }
    }
});
```

NodeJS

Exercice

- En utilisant les deux fonctions `readFile` et `writeFile` du module `file system`, écrire un programme NodeJS qui permet de copier le contenu du fichier salutation.txt dans un nouveau fichier ”salutation - copy .txt”

NodeJS

Correction

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

fs.readFile("salutation.txt", "utf8", function (err,
  result) {
  if (err) return console.error(err);

  fs.writeFile("salutation␣-␣copy.txt", result,
   function (err) {
    if (err) return console.error(err);
    console.log("C'est␣fait");
  });
});
```

NodeJS

Qu'affiche le programme suivant ?

```
process.on("beforeExit", function() {
  console.log("beforeExit");
});

console.log("Begin");
setTimeout(function () {
  console.log("the asynchronous timeout");
  console.log("End");
}, 0);
```

NodeJS

Qu'affiche le programme suivant ?

```
process.on("beforeExit", function() {
  console.log("beforeExit");
});

console.log("Begin");
setTimeout(function () {
  console.log("the asynchronous timeout");
  console.log("End");
}, 0);
```

Résultat

```
Begin
the asynchronous timeout
End
beforeExit
```

NodeJS

Qu'affiche le programme suivant ?

```
process.on("beforeExit", function() {
  console.log("Last call before end");
  setTimeout(function () {
    console.log("Good bye");
  }, 500);
});

console.log("Begin");
setTimeout(function () {
  console.log("the asynchronous timeout");
  console.log("End");
}, 0);
```

NodeJS

Qu'affiche le programme suivant ?

```
process.on("beforeExit", function() {
  console.log("Last call before end");
});

process.on("exit", function(code) {
  console.log("The end of the program :", code);
});
console.log("Begin");
setTimeout(function () {
  console.log("the asynchronous timeout");
  console.log("End");
}, 0);
```

NodeJS

Qu'affiche le programme suivant ?

```
process.on("beforeExit", function() {
  console.log("Last call before end");
  setTimeout(function () {
    console.log("Good bye");
  }, 500);
});

process.on("exit", function(code) {
  console.log("The end of the program :", code);
});
console.log("Begin");
setTimeout(function () {
  console.log("the asynchronous timeout");
  console.log("End");
}, 0);
```

NodeJS

Qu'affiche le programme suivant ?

```
process.on("beforeExit", function() {
  console.log("Last call before end");
  setTimeout(function () {
    console.log("Good bye");
  }, 500);
});

process.on("exit", function(code) {
  console.log("The end of the program :", code);
});
console.log("Begin");
setTimeout(function () {
  console.log("the asynchronous timeout");
  console.log("End");
}, 0);
```

NodeJS

Qu'affiche le programme suivant ? pourquoi ?

```
process.on('uncaughtException', function (err) {
  console.log('Exception␣captee:␣␣' + err);
});

setInterval(function () {
  console.log('ca␣marche');
}, 500);

var obj = {a : 5, b : 3};
obj.c.d;
console.log('ca␣ne␣marche␣pas');

console.log('la␣suite');
```

NodeJS

Que fait le programme suivant ?

```
process.stdin.resume();

process.on('SIGINT', function() {
  console.log('SIGINT␣Attrapee.');
});
```

NodeJS

Utiliser le module os pour afficher :

- L'architecture de votre machine
- Le nombre de CPU
- Le hostname
- Et la charge moyenne

NodeJS

Correction

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

console.log("Architecture:", os.arch());
console.log("Nombre CPU", os.cpus().length);
console.log("Hostname", os.hostname());
console.log("La charge moyenne", os.loadavg());
```

NodeJS

Télécharger le module `colors` ensuite tester le programme suivant ?

```
require('colors');

console.log('Attention!'.red);
console.log("c'est␣correct!".green.underline);
console.log('celui-ci␣aussi'.bgGreen.white.bold);
console.log('lettres␣en␣couleurs'.rainbow);
console.log('drapeau␣americain␣et␣francais'.america)
 ;
console.log("c'est␣quoi␣ce␣truc".trap);
```

NodeJS

Créons notre serveur

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

var server = http.createServer(function(req, res) {
  res.writeHead(200);
  res.end('Hello world ');
});
server.listen(8080);
```

- On crée un serveur qui attend les clients sur le port 8080 (port généralement utilisé par nodeJS)
- À la connexion d'un client, le serveur affiche hello world
- `writeHead` permet d'écrire à l'entête de la réponse l'état 200 : tout fonctionne bien

NodeJS

Comment retourner du code HTML ?

```
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>Hello␣world</p>'+
'␣␣␣␣</body>'+
'</html>');
  res.end();
});
server.listen(8080);
```

NodeJS

Que fait le programme suivant ? connectez-vous au serveur avec des url différentes : localhost :8080, localhost8080/mapage...

```
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('Hello world, this is your requested
     page : ' + page);
    res.end();
});

server.listen(8080);
```

NodeJS

Comment on fait pour récupérer les paramètres d'une requête ?

```
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 ('prenom' in params && 'nom' in params) {
        res.write('Vous␣etes␣' + params['prenom'] + '␣' +
          params['nom']);
    }
    else {
        res.write('Vous␣devez␣bien␣avoir␣un␣prenom␣et␣un␣
         nom,␣non␣?');
    }
    res.end();
});
server.listen(8080);
```

NodeJS

Exercice

- Ecrire un programme qui affiche le résultat d'une opération arithmétique des nombres passés en paramètre

NodeJS

Correction

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

function calcul(tab, operator) {
        var result = '';
        for(var i in tab){
                result = result + operator + tab[i];
        }

        return eval(result.substr(1));
}
```

Suite

```
var server = http.createServer(function(req, res) {
        var pathname = url.parse(req.url).pathname;
        var params = querystring.parse(url.parse(req.url).query);
        res.writeHead(200, {'Content-type': 'text/plain'});

        var result;

        if(pathname === '/addition'){
                result = calcul(params, '+');

        } else if(pathname === '/soustraction'){
                result = calcul(params, '-');

        } else if(pathname === '/multiplication'){
                result = calcul(params, '*');

        } else if(pathname === '/division'){
                result = calcul(params, '/');
        }

        res.write('Resulat : ' + result);
        res.end();
});
server.listen(8081);
```

NodeJS

Exercice

- Ecrire un programme qui génère dans un fichier la table de multiplication, addition, soustraction ou division du nombre passé en paramètre

NodeJS

Correction

```
let http = require('http');
let url = require('url');
let querystring = require('querystring');
let fs = require('fs');
function calculTable(params, operator) {
  let t = '';
  for(let i in params){
    t += 'Table de '+params[i]+'\n';
    for(let j = 0; j <= 10; j++){
      t += params[i]+operator+j+' = '+eval(params[i]+operator+j
       )+'\n';
    }
    t += '\n\n----------\n\n';
  }
  console.log(t);
  return t;
}
```

Suite

```
let server = http.createServer(function(req, res) {
        let pathname = url.parse(req.url).pathname;
        let params = querystring.parse(url.parse(req.url).query);
        res.writeHead(200, {'Content-type': 'text/html'});
        let table = '';
        if(pathname === '/addition'){
                table = calculTable(params, '+');

        }else if(pathname === '/soustraction'){
                table = calculTable(params, '-');

        }else if(pathname === '/multiplication'){
                table = calculTable(params, '*');

        }else if(pathname === '/division'){
                table = calculTable(params, '/');

        }else if(pathname === '/assets/table.txt'){
                let t = fs.readFileSync('./assets/table.txt');
                res.write(t);
        }
```

Suite

```
        if(table !== ''){
                fs.writeFile('./assets/table.txt', table, (err) => {
                        if(err){
                                console.log(err);
                        }
                });
                res.write(
                        `<!DOCTYPE html>
                <html lang="en">
                <head>
                        <meta charset="UTF-8">
                        <title>Page nodeJS</title>
                </head>
                        <body>
                        <p><a href="/assets/table.txt" download="/
                         assets/table.txt">Cliquez ici pour afficher le
                          fichier</a></p>
                        </body>
                </html>`);
        }
        res.end();
});
server.listen(8081);
```

NodeJS

Exercice

- Ecrire un programme qui selon les paramètres de la requête get (option = c ou a), il crée ou ajoute du contenu (passé en paramètre) au fichier qui correspond au pathname de la requête.

NodeJS

Correction

```
const http = require('http');
const url  = require('url');
const querystring = require('querystring');
const fs = require('fs');

let server = http.createServer((req,res)=>{
    let path = url.parse(req.url).pathname;
    let params = querystring.parse(url.parse(req.url).query);
    res.writeHead(200,{"Content-type":"text/html"});
    console.log(params);
    if (fs.existsSync(`${path.substr(1)}.txt`)){
        if (params.option === "a")
            fs.appendFileSync(`${path.substr(1)}.txt`, `${
             params.content}\n`);
        else
            fs.writeFile(`${path.substr(1)}.txt`, `${params.
             content}\n`, (err) => {
```

NodeJS

Suite

```
                if (err) {
                    console.log(err);
                }
            });
    }
    else{
            fs.writeFile(`${path.substr(1)}.txt`, `${params.
             content}\n`, (err) => {
                if (err) {
                    console.log(err);
                }
            });
    }
    res.end();
});
server.listen(8085);
```

NodeJS

Express

- est un module nodeJS particulier
- est un framework permettant de créer des applications nodeJS
- plus de détails dans le cours Express