Programmation Web – Avancé

JavaScript & Node.js

Introduction à Node.js

Introduction à Node.js

Créons des applications côté serveur en JavaScript...

Introduction à Node.js

- Environnement serveur open source
- Création d'outils et applications côté serveur en JS
- Utilisation optimale des ressources des serveurs sans dépendance à un serveur http externe
- Multiplateforme (Windows, Linux, Mac...).

JS côté serveur : quel Software utiliser pour développer ?

- Installation de l'environnement **Node.js** LTS [54.]
- Pour écrire et exécuter vos scripts :
  - **VS Code**
  - N'importe quel terminal : **Git Bash**
- Pour gérer différentes versions de votre code et différentes machines : **Git, Gitlab / Github** et **VS Code**

JS côté serveur : quel Software utiliser pour développer ?

- Rapide consommation / tests de Web APIs : **REST Client** [77.] comme extension de **VS Code**

Exécution et déploiement d'applications web sur le cloud : quels services ?

- Heroku ou MS Azure ou AWS ou Netlify ou...

JS côté serveur : Où mettre votre code Node.js ?

A. Directement via l'interpréteur de commandes de Node au sein d'un terminal : **node**
   - PRESENTATION : vérifier l'installation de Node.js & opérations mathématiques de base

JS côté serveur : Où mettre votre code Node.js ?

B. Via l'interpréteur de commandes de Node au sein d'un terminal et d'un script externe :
   - Start : **node script_name** (.js optionnel)
   - Stop : **CTRL + c**

- **DEMO** : Appel d'une méthode au sein d'un script externe via Node.

Introduction aux modules sous Node.js

- Un module = librairie JS fournissant des fonctions et / ou des objets
- Création d'un module :
  - Création d'un fichier **nomModule.js**
  - Au sein du fichier **nomModule.js**, utilisation de **module.exports** ou **exports** pour rendre les propriétés et méthodes disponibles en dehors du fichier module

Exporter des objets en Node.js

- Export à la fin d'un script

```
module.exports = {authorize, users };
```

- Export à la fin d'un script : équivalent d'un default export en ES6

```
module.exports = router;
```

Exporter des objets en Node.js

- Export « à la volée »

```
module.exports.authorize = authorize;
module.exports.users = users ;
```

- Export « à la volée », version plus courte

```
exports.authorize = authorize;
exports.users = users ;
```

Importer un module sous Node.js

- Inclure un module intégré ou installé :

```
// module integrated to the runtime environment
let http = require("http");
// module following package installation
let shortid = require("shortid");
```

Importer un module sous Node.js

- Inclure un propre module :

```
var pizzaRouter = require('./routes/pizzas’);

const { users, authorize } = require("../utils/auths");
```

Précisions

- **module.exports** : référence de l'objet retournée par l'appel de **required()**
- **exports** : référence vers **module.exports**, **exports** non retourné par l'appel de **required()**

- Mauvais export 

```
exports = { authorize, users }; /* exports has a new reference,
                                   it is no longer linked to module.exports */
```

Node.js & utilisation de modules intégrés

- **DEMO :** Serveur de fichiers statiques minimaliste avec Node.js fonctionnant sur le port *7777* et mettant à disposition les fichiers présents dans le répertoire **/public**
  - « Parser » une URL dans ses parties : **new URL(request.url, `http://${request.headers.host}`);**
  - Lecture de la propriété « chemin » données dans un objet URL (myURL) : **myURL.pathname**
  - Lecture asynchrone d'un fichier : **fs.readFile()**
  - Chemin absolu où Node.js est appelé : **__dirname**

Introduction aux packages sous Node.js

- Tous les fichiers nécessaires pour un module sont contenus dans un package
- Installer un package (via terminal) : **npm install package_name**
- NPM : gestionnaire de package [55.]

Introduction aux packages sous Node.js

- Packages associés à une app : **package.json**
  - Création sur base de questions ou par défaut de **package.json** : **npm init** ou **npm init –y**

```
{ "name": "node-modules",
  "dependencies": {
    "mime": "^2.4.6",
    "shortid": "^2.2.15"
  },
  "devDependencies": {},
  "scripts": {
    "start": "node node-modules.js"
  },
  "author": "e-Baron"}
```

Introduction aux packages sous Node.js

- Arbre exact des dépendances installées : package-lock.json
  - Génération automatique pour chaque opération modifiant node_modules ou package.json

```
{
  "requires": true,
  "lockfileVersion": 1,
  "dependencies": {
    "mime": {
      "version": "2.4.6",
      "resolved": "https://registry.npmjs.org/mime/-/mime-2.4.6.tgz",
      …
```

Introduction aux packages sous Node.js

- Localisation d'un module par Node : recherche parmis tous les chemins specifies dans **module.paths** (**node_modules**...)
- Mise à jour des packages vers leur dernière version [102.]

Introduction aux packages sous Node.js

- **DEMO** : Serveur de fichiers statiques minimaliste avec Node.js, des modules et des packages
  - Génération aléatoire d'un ID à l'aide d'un package NPM : **uuid**
  - Gestion du MIME type à l'aide du package **mime**
  - Export & import du module **ServerConfiguration**

Accélérons notre développement

- Utilisation d’un framework