# Présentation

- Node.JS est une plateforme de développement et d'exécution d'applications client serveur en Javascript Créé en 2009, le noyau de Node.JS est basé sur V8 le moteur Javascript de chrome et la librairie libuv (E/S asynchrones).
- Node.JS supporte le modèle de programmation asynchrone/événementielle.
- Node.JS se distingue par la réduction du temps de traitement et la capacité à traiter de nombreuses requêtes simultanément.
- Node.JS implémente le standard commonJS pour supporter le développement d'applications modulaires
- Installation
  - Site officiel: https://nodejs.org/en/download/
  - NVM (Node Version Manager) : outil de gestion des versions pour node.

# NPM (Node Package Manager)

- Outil pour télécharger, installer et gérer des modules Node.JS, la majorité des modules sont hébergés sur github.
- Le fichier package.json une description des modules utilisés par une application.
- Npm supporte deux modes d'installation des modules:
  - Local: le module est installé le dossier node_modules du dossier courant, qui doit être le dossier de l'application où le module sera installé (npm install nom-module)
  - Global: le module est installé globalement dans le dossier %AppData%/npm/node_modules pour windows (nom install –g nom-module) .

# Le fichier package.json

- Le fichier package.json situé à la racine d'un projet Node.JS contient les dépendances de l'application, le fichier contient aussi des informations sur l'application comme le nom, la version, les auteurs…etc.
- Pour créer ce fichier, il suffit de lancer la commande npm init dans le dossier de l'application.
- Pour que NPM puisse ajouter une dépendance automatiquement, il suffit d'indiquer l'option --save au moment de l'installation.
- Si le fichier package.json contient les dépendances de l'application, il suffit d'exécuter la commande npm install(sans paramètres) à la racine du répertoire où se trouve le fichier package.json, NPM va alors lire automatiquement ce fichier puis installera toutes les dépendances qui y sont indiquées.

# Le module fs (FileSystem)

## Ecriture dans un fichier

```
//Ecrire dans un fichier text en mode asynchrone.
var fs = require('fs');
fs.writeFile('data.txt',
    'création d\'un fichier!',
    err => {
if(err) { throw err; }
console.log('Le fichier est enregisté!');
});
```

## Lecture d'un fichier

```
//Lecture

    fs.readFile('data.txt', (err, data)=> {
if (err) throw err;
console.log(data.toString());
});
```

# Le module child_process

## Exécution d'un processus

```
    var processus=require('child_process');
processus.exec('dir /w', (err,stdout,stderr)=>{
    console.log('stdout:'+stdout);
    console.log('stderr:'+stderr);
    if (err!=null)
    {
        console.log('Erreur:' + err);
    }
});
```

# Le module url

```
//app2.js
var http=require('http');
var url=require ('url');

//Créer un serveur web
var serveur=http.createServer(
    //Traiter les requêtes

    (req,rep)=>
    {   var chemin =url.parse(req.url).pathname;

        rep.end("<h2>Un serveur web</h2>" +
            "fichier demandé:" + chemin);

    }
    );


//Démarrage du serveur
serveur.listen(4654);
```

# Les modules

- Une applications node.js peut être composée de plusieurs, blocs de fonctionnalités distinctes appelées “modules”.
- Node implémente le standard CommonJS, qui permet de gérer les modules en JavaScript.
- Dans Node un module peut être référencé soit par son nom soit par son chemin.

# Création d'un module

Création du module module1.js

```
//module1.js

exports.titre='Un exemple de module';
exports.executer=()=>{console.log("Un exemple de fonction dans un module");};
```

- module.exports permet d'exposer l'API du module

Utilisation du module

```
1  //testModule1.js
2  var mod1=require("./module1");
3  console.log("Titre:" + mod1.titre);
4  mod1.executer();
5
```

# Exemple 1: Un serveur Web

```
var http=require('http');
var serveur=http.createServer(
//Traiter le requête http
(req,rep)=>{
    var body="Serveur Web\n";
    var contentLengh=body.length;
    //Entête de la réponse HTTP
    rep.writeHead(200,{
        'Content-Type':'text/plain',
        'Content-length':contentLengh
    });

    //Envoi de la réponse.
    rep.end(body);
}
);

//Démarrer le serveur
serveur.listen(7878);
```

- Req est de type RequestServer, et rep est de type ResponseServer)
- rep.writeHead accepte deux paramètres, le premier est le code du statut de la requête, les autres entêtes de la réponse HTTP sont fournis dans un objet JSON en deuxième paramètres.

```
C:\Users\nadir>curl -i http://localhost:7878
HTTP/1.1 200 OK
Content-Type: text/plain
Content-length: 12
Date: Fri, 25 Dec 2015 17:24:38 GMT
Connection: keep-alive

Serveur Web
```

- Lien de téléchargement CURL: http://www.confusedbycode.com/curl/

# Le mode debug

- Activer le mode debug: Node debug script.js
- Commandes
  - cont: continuer l'exécuter
  - next: exécuter l'instruction suviant
  - step:pas à pas détaillé.
  - out : continuer l'exécution jusqu' à la fin de la fonction en cours
  - backtrace: afficher la liste des appels
  - repl (Real Eval Print Loop): démarrer le mode repl
  - watch(expr)
  - list(n): lister n lignes de code à partir de la ligne en cours

# Exemple  Serveur Web

```
var http = require('http');
var url = require('url');
var fs = require('fs');
var serveur = http.createServer(
    //Traiter le requête http
    (req, rep) => {
        var chemin = url.parse(req.url).pathname;
        //Journalisation
        console.log("Chargement:" + chemin);
        var racine = './www';
        var type = '';
        var contenu = '';
        type = getType(chemin);
        fichier = racine + chemin;
        fs.readFile(fichier, (err, data) => {
            if (err != null) {
                console.log("Erreur : " + err);
                contenu = 'Erreur:' + err;
                rep.writeHead(404, {
                    'Content-Type': 'text/plain',
                    'Content-length': contenu.length
                });

                //Envoi de la réponse.
                rep.end(contenu);

            } else {
                contenu = data;
                var contentLengh = contenu.length;
                //Entête de la réponse HTTP
                rep.writeHead(200, {
                    'Content-Type': type,
                    'Content-length': contentLengh
                });

                //Envoi de la réponse.
                rep.end(contenu);
            } //fin else
        }); //Fin ReadFile
    }
); //Fin create Server


function getType(chemin) {
    var extension = chemin.split('.').pop();
    switch (extension) {
        case 'html':
            return 'text/html';
        case 'css':
            return 'text/css';
        case 'js':
            return 'application/javascript';
        case 'json':
            return 'application/json';
        case 'jpg':
            return 'image/jpeg';
        case 'jpeg':
            return 'image/jpeg';
        case 'png':
            return 'image/png';
        case 'ico':
            return 'image/ico';
        default:
            return 'text/plain'

    }
}
//Démarrer le serveur
serveur.listen(7878);
```

# Les frameworks

- Express: un framework minimaliste de développement d'applications web.
- Mean.js: développement web avec bases de de données NoSQL (Express, MongoDB, Angular, Grunt pour l'automatisation des tests.
- Hapi: framework de développement d'application (validation des entrées, cache, gestion des erreurs, journalisation)
- Socket.io: développement d'application temps réel
- Meteor: framework MVC de développement d'application web et mobiles
- Sails.js : framework MVC de développement d'applications

# Utiliser MongoDb dans nodeJs

- Installation du driver MongoDB
- Connexion

```
mongo-express>npm install mongodb --save
```

```
var mongo = require('mongodb');
var uri = "mongodb://localhost:27017";
mongo.MongoClient.connect(uri, (err, dbServer) => {
    … Traitements des données
        dbServer est une référence du serveur mongodb…
        err contient le message d'erreur dans le cas d'une
        erreur de connexion, sinon il est null.
});
```

# Ajout d'un document dans une collection

```
var doc = {
        titre : 'De la démocratie en amérique',
        année : 1888,
        auteur : 'Alexis de Tocqueville',
        prix : 4566.33,
        editeurs : ['Flamarion', 'Ellipse'],
        note: {
            critiques: 34,
            audience: 78

        }
    }
   let db = dbServer.db("biblio");
   db.collection('livres').insert(doc, (err, res) => {
        if (err) {
            console.log(err);
            process.exit(1);
        }
          console.log(res);

    });
```

```
{ result: { ok: 1, n: 1 },
  ops:
   [ { titre: 'De la démocratie en amérique',
       'année': 1888,
       auteur: 'Alexis de Tocqueville',
       prix: 4566.33,
       editeurs: [Object],
       note: [Object],
       _id: 5696d14543369ab0060fb5a3 } ],
  insertedCount: 1,
  insertedIds: [ 5696d14543369ab0060fb5a3 ] }
```

# Interrogation des données

- Exemples de requêtes

```
var  requete={année:1897}
requete={editeur:'Flamarion'}
requete={'note.critiques': {'$gte':69}}
```

- Exécution de la requête et traitement du résultat:

```
db.collection(livres').find(requete).toArray((err, docs) =>
{
        if (err) {
            console.log(err);
            process.exit(1);
        }
        console.log('Documents trouvés:');
        docs.forEach(function (doc) {
console.log(JSON.stringify(doc)); });
        process.exit(0);
    });
```

- https://docs.mongodb.org/manual/tutorial/query-documents/

# Mongoose :ODM (Object Document Mapper)

- Installation: `\mongo-express>npm install mongoose --save`

- Définition du schéma de la base de données

```
//schema.js
var mongoose = require('mongoose');
//Définition du schéma de la base de données biblio
module.exports = new mongoose.Schema({
titre: { type: String  },
annee: {type:Number ,Default: Date.now},
prix: { type: Number, min:10,max:1200},
auteur:{type: String},
editeurs: [String],
note:{critique: { type: Number },audience:{ type: Number}}
});
```

http://mongoosejs.com/docs/3.6.x/docs/guide.html

# Utiliser Mongoose

1. Ajouter un livre

```
//test-mongoose.js

var mongoose = require('mongoose');
var schema = require('./schema.js');
mongoose.connect('mongodb://localhost:27017/test');

//Paramètres:nom modèle,schéma, nom collection.
var Livre = mongoose.model('Livre', schema, 'livres');
var livre = new Livre({
    titre: 'Discours de la méthode',
    annee: 1637,
    auteur: 'René Descartes',
    prix: 150,
    editeurs: ['Flamarion', 'Librio'],
    note: {
        critiques: 35,
        audience: 122

    }
});
livre.save(function(err, res) {
    if (err) {
        console.log(err);
        process.exit(1);
    }
    console.log(res);

});
```

```
D:\FO2011\Node\atl\cours\form_express_rest>node test-mongoose
{ editeurs: [ 'Flamarion', 'Librio' ],
  note: { audience: 122 },
  _id: 5696e11365b1c22416181bf6,
  prix: 150,
  auteur: 'Ren□ Descartes',
  annee: 1637,
  titre: 'Discours de la m□thode',
  __v: 0 }
```

- Recherche

```
Livre.find({ auteur: 'René Descartes' }, function (err, docs) {
    if (err) {
        console.log(err);
        process.exit(1);
    }
    console.log(require('util').inspect(docs));
    process.exit(0);
});
```

# Application: API REST

- Objectif: réaliser une api REST avec Mongoose, NodeJS

| URI | GET | POST | PUT | DELETE |
|---|---|---|---|---|
| /ressources | Retourner toutes les ressources | Ajouter une ressource | | |
| /ressource/id | Retourne une ressource | | Modifier une ressource | Supprimer une ressource |