

Notes de cours :



The screenshot shows a dark-themed user interface for a learning platform. On the left, there's a decorative header featuring the Node.js logo and several white icons representing web development: a globe, a cloud, a database, and a code editor. To the right of the header, the text "Apprendre Node.js & Créer une API REST de A à Z !" is displayed. Below this, a red button says "Continuer : Session 1". Underneath the button is a rating section with five yellow stars and the text "Modifier votre note". At the bottom of the screenshot, a progress bar indicates "106 éléments terminés sur 106" and a link to "Réinitialiser la progression".

par

Francis Poissant
(Enseignant au secondaire et apprenti-codeur!)
francispoissant@gmail.com

Participant au cours de Bryan en janvier 2019

Notes réalisées avec ActivInspire, un logiciel pour tableau blanc interactif

Parce que la mémoire est une faculté qui oublie!

Précision:

- Ce sont des notes personnelles
- Elles commencent avec la section 3
- Elles deviennent plus systématiques avec la vidéo 23
- Mais pas forcément tout le temps!



Pour comprendre Node.js il faut d'abord comprendre le Javascript

Code JS → Moteur JS → Code machine

Chaque navigateur a son propre moteur JS

Chakra V8 SpiderMonkey

Fonctionnement interne

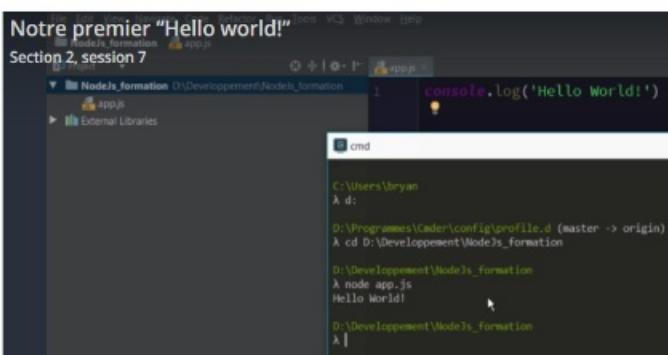
Section 3, session 11

Javascript est conçu pour gérer les événements, donc Node.js a pu mettre en place un environnement sans commande bloquante.

C'est à dire qu'il gère l'asynchrone, il peut donc faire plusieurs choses à la fois.

<u>Commande bloquante</u>	<u>Commande non-bloquante</u>
1) Télécharger un fichier	1) Télécharger un fichier
2) Afficher le fichier	3) Dès que c'est terminé, afficher le fichier
3) Faire autre chose	2) Faire autre chose

Exécuter un fichier avec node.js: **node app.js**



node -v : version de node.js; **node --help**: les commandes node.js

Points forts

Entreprises

- Microsoft
- Applications en temps réel
- LinkedIn
- Applications scalables
- PayPal
- Grosse communauté
- Très nombreuses librairies (modules)

Ryan Dahl

- Moteur V8
- Application C++
- Node
- Modules internes

installer nodemon:

```
D:\Développement\Node.js_formation
  npm install -g nodemon
  [.....] - rollbackFailedOptional: verb npm
```

Nodemon exécute automatiquement le code dans terminal

nodemon app.js pour démarrer
rs en console pour rafraîchir

Utilisation d'ECMAScript6 (ES6)

Section 4

```
D:\Développement\NodeJs_formation
λ npm install --save babel-register
```

Babel est maintenant inutile avec node.js

```
app.js ×
1  require('babel-register');
2  console.log('test')
```

```
C:\Users\Don-de-Dieu\Desktop\node-formation>node app.js
test
```

scope dans les blocs ({ entre accolades })

Let & const : déclaration de variables
Section 4, session 14

let pour variable
const pour des variables constantes

Section 5:

Le système des modules

Un module

- Fichier
- Syntaxe particulière
- Puissance de Node.js

modules internes à node:

require('os') = pour l'os (operating system)
require('fs') = file system
require('http')

```
app.js  test.txt ×
1  const os = require('os');
2  console.log(os.arch());
3  console.log(os.homedir());
4
5  const fs = require('fs');
6
7  fs.readFile('test.txt', 'utf-8', (err, data) => {
8    if (err) {
9      console.log(err)
10    } else {
11      console.log(data)
12      fs.writeFile('test.txt', 'Hello World', 'utf-8', (err) => {
13        if (err) {
14          fs.readFile('test.txt', 'utf-8', (err, data) => {
15            console.log(data)
16          })
17        }
18      })
19    }
})
```

// x64
// c:\Users\Don-de-dieu
(voir node.js api reference
documentation os)

// lit le fichier (fs.readFile())

// écrit dans le fichier fs.writeFile()

Module http: req =requête res=response http.createServer res.writeHead res.write res.end

The screenshot shows a Node.js development environment with two tabs open: 'app.js' and 'package-lock.json'. The 'app.js' tab contains the following code:

```
1 const http = require('http');
2
3 http.createServer((req, res) => {
4     if (req.url === '/') {
5         res.writeHead(200, {'Content-type': 'text/html'});
6         res.write("<h1>Accueil</h1>\n");
7     } else {
8         res.writeHead(404, {'Content-type': 'text/html'});
9         res.write("<span style='color: red'>Erreur 404</span>");
10    }
11
12    res.end();
13
14 }).listen(8080);
15
```

Below the code editor are two browser windows. The left window shows the URL `localhost:8080/qqch` and displays the error message "Erreur 404". The right window shows the URL `localhost:8080/` and displays the page title "Accueil".

Erreur 404

Accueil

23 - Combinaison de modules

The screenshot shows a Node.js development environment with three tabs open: 'app.js', 'package-lock.json', and 'test.txt'. The 'app.js' tab contains the following code:

```
1 const http = require('http');
2 const fs = require('fs');
3
4 http.createServer((req, res) => {
5     if (req.url === '/') {
6         res.writeHead(200, {'Content-type': 'text/html'});
7         res.write("<h1>Accueil</h1>\n");
8         res.end();
9     } else if (req.url === '/test') {
10        fs.readFile('test.txt', 'utf-8', (err, data) => {
11            if (err) {
12                send404(res);
13            } else {
14                res.writeHead(200, {'Content-type': 'text/html'});
15                res.write(data);
16                res.end();
17            }
18        });
19    } else {
20        send404(res);
21    }
22 }).listen(8080);
23
24 function send404(res) {
25     res.writeHead(404, {'Content-type': 'text/html'});
26     res.write("<span style='color: red'>Erreur 404</span>");
27     res.end();
28 }
```

Ici, on a deux if router: /test renvoie le contenu de test.txt

res.end a dû être déplacé et doit être répété, car par exemple le `readFile` envoie une réponse asynchrone, ce qui fait planter l'app si on ferme la réponse dans le champ global.

On a créé la fonction `send404(res)` pour optimiser le code.

24-faire ses modules

```
Project node-formation C:\Users\Don-de-Dieu\VS Code\app1.js module1.js
node-formation C:\Users\Don-de-Dieu\VS Code\app1.js module1.js
1 exports.sayHello = function() {
2     console.log('Allo !')
3 }
4 exports.sayHi = function() {
5     console.log('Hi !')
6 }
7 exports.hello = "Hello World !"
```

```
Project node-formation C:\Users\Don-de-Dieu\VS Code\app1.js module1.js
node-formation C:\Users\Don-de-Dieu\VS Code\app1.js module1.js
1 const mod1 = require('./module1'); // ici on peut enlever ./ devant module1
2 // si on place ses modules dans un dossier node_modules (convention de node.js)
3
4 mod1.sayHello(); // méthode 1
5 mod1.sayHi(); // méthode 2
6 console.log(mod1.hello); // variable hello
```

N.b.: Mon dossier node_modules est plein de modules pour babel... en ce moment.
J'ai laissé module1 dans le niveau hiérarchique de app.js pour ne pas le perdre!

Les modules via NPM section 6

NPM

- npmjs.org
- Node Package Manager
- Command line
- « apt-get » de Linux

```
package.json
1 {
2     "name": "nodejs_formation",
3     "version": "1.0.0",
4     "description": "",
5     "main": "app.js",
6     "scripts": {
7         "test": "echo \"Error: no test specified\" && exit 1",
8         "start": "nodemon app.js"
9     },
10    "author": "",
11    "license": "ISC"
12 }
```

```
npm
Use `npm install <pkg>` afterwards to install a package and save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (nodejs_formation)
version: (1.0.0)
description:
entry point: (index.js) app.js
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to D:\Développement\NodeJs_formation\package.json:

{
    "name": "nodejs_formation",
    "version": "1.0.0",
    "description": "",
    "main": "app.js",
    "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "author": "",
    "license": "ISC"
}

Is this OK? (yes) |
```

npm run test: exécutera la commande dans "scripts" de package.json

On peut ajouter une commande comme "start": "nodemon app.js" (**npm run start**)

ou juste: **npm start**

Lors de l'intallation des modules, toutes les dépendances du projet seront notées dans ce package.json.

Installation de module avec npm

ex. avec babel-register: sur npmjs.com, on aura les instructions d'installation, "dépendances" qui est tout ce que Babel installe et les "dépendants" qui installent babel-register si on les installe.

npm i babel-register plus besoin de mettre --save qui permettait d'enregistrer le package dans package.json (se fait maintenant automatiquement)

version: "^6.26.0" => la plus récente: 6.x.x

"~6.26.0" => la plus récente: 6.26.x

"6.26.0" => précisément cette version

```
license": "ISC",
"dependencies": {
    "babel-register": "^6.26.0"
}
```

npm list: toutes les dépendances ; **npm list --depth=0** (directe); **npm list --depth=1** (avec niveau 1)

NPM suite:

Infos sur un module: npm view (nom du module)

```
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
λ npm view babel-register
```

```
babel-register@6.26.0 | MIT | deps: 7 | versions: 50
babel require hook
```

```
dist
.tarball https://registry.npmjs.org/babel-register/-/babel-register-6.26.0.tgz
.shasum 0ed021173e2fcb486d7acb45c6009a856f64701
```

```
dependencies:
babel-core: ^6.26.0      core-js: ^2.5.0      lodash: ^4.17.4      source-map-support: ^0.4.15
babel-runtime: ^6.26.0     home-or-tmp: ^2.0.0     mkdirp: ^0.5.1
```

```
maintainers:
- existentialism <dhng12@gmail.com>
- thejameskyle <me@thejameskyle.com>
- sebmck <sebmck@gmail.com>
- danez <daniel@tschinder.de>
- hzoo <hi@henryzoo.com>
- logansmyth <loganfsmyth@gmail.com>
```

```
dist-tags:
latest: 6.26.0    next: 7.0.0-beta.3
```

```
published 10 months ago by hzoo <hi@henryzoo.com>
```

```
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
```

```
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
λ npm view babel-register dependencies
```

```
{ 'babel-core': '^6.26.0',
'babel-runtime': '^6.26.0',
'core-js': '^2.5.0',
'home-or-tmp': '^2.0.0',
lodash: '^4.17.4',
mkdirp: '^0.5.1',
'source-map-support': '^0.4.15' }
```

```
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
λ npm view babel-register dependencies
```

```
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
λ npm view babel-register versions
```

```
[ '6.1.4',
'6.1.17',
'6.1.18',
'6.2.0',
'6.2.4',
'6.3.2',
'6.3.13',
'6.4.3',
'6.5.0-1',
'6.5.0',
'6.5.1' ]
```

```
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
λ npm view babel-register maintainers
```

```
[ 'thejameskyle <me@thejameskyle.com>,
'sebmck <sebmck@gmail.com>,
'danez <daniel@tschinder.de>,
'hzoo <hi@henryzoo.com>,
'loganfsmyth <loganfsmyth@gmail.com>' ]
```

```
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
```

Installer une version particulière:

```
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
λ npm install babel-register@6.1.4
[ ..... ] - fetchMetadata: sill resolveWithNewModule
```

Mise à jour:

```
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
λ npm outdated
```

babel-register	6.1.4	6.26.0	6.26.0	nodejs_formation
----------------	-------	--------	--------	------------------

```
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
```

Après: npm update : va faire la mise à jour

Désinstaller un module:

```
npm uninstall babel-register ou npm un babel-register
```

Comprendre les devDependencies: modules utiles pour le développement seulement.

ex.: ms qui permet de transformer des unités de temps en secondes

```
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
λ npm install ms --save-dev
npm WARN nodejs_formation@1.0.0 No description
npm WARN nodejs_formation@1.0.0 No repository field.
[ ..... ] | postinstall: [WARN] nodejs_formation@1.0.0
```

```
  babel-register . "6.26.0"
  },
  "devDependencies": {
    "ms": "^2.1.1"
  }
}
```

```
app.js  package.json
1  var ms = require('ms')
2  console.log(ms('2 days'))
```

```
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
λ node app.js
17280000
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
```

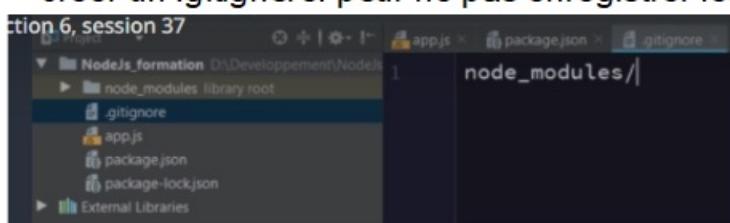
36. Comprendre le modules globaux (-g)

```
D:\Développement\NodeJs_formation (nodejs_formation@1.0.0)
λ npm install -g nodemon
```

⇒ dispo partout sur l'ordi

37. Gérer git & npm sur un même projet

créer un .gitignore: pour ne pas enregistrer les dépendances (inutiles)



```
node_modules/
.idea/|
```

.idea/ est pour php storm qui crée ce fichier

Comprendre une API REST

39. Définition d'une API REST



REST

→ Representational state transfer

→ Architecture de requêtes i.e. de liaisons entre client et serveur

→ RESTful Pour parler de cette architecture

→ API



Requêtes (CRUD)

→ Create

Commandes normales ou courantes faites à une base de données

→ Read

→ Update

→ Delete

40. Les méthodes du protocole HTTP l'api rest va utiliser ces méthodes pour faire la liaison entre le client et le serveur

Méthodes HTTP

- GET
- POST
- PUT
- DELETE
- PATCH
- HEAD
- TRACE
- OPTIONS
- CONNECT

Opération	HTTP	SQL
Create	POST	INSERT
Read	GET	SELECT
Update	PUT	UPDATE
Delete	DELETE	DELETE

Requête

GET

https://bestbooks.com/api/v2/books/34

GET

https://bestbooks.com/api/v2/books

Réponse en json

{
id: 34,
name: "100 raisons d'apprendre Node.js",
date: "01/07/2018"
}

array d'objets

[
{
id: 34,
name: "100 raisons d'apprendre Node.js",
date: "01/07/2018"
},
{...}
]

Requête

POST

```
https://bestbooks.com/api/v2/books
{
  name: "10 raisons de noter un cours"
}
```

Réponse

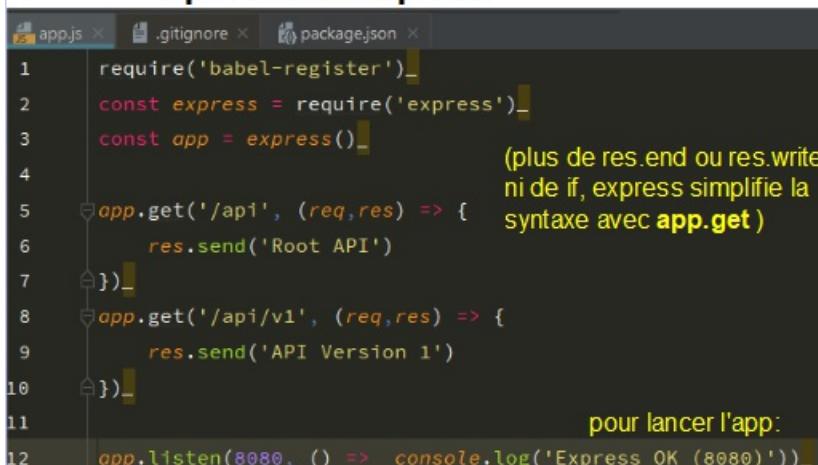
```
{
  success: true
  id: 65
}
```

Section 8: Express - Crédit d'une API REST

Installation de express qui va gérer les url

voir doc: <http://expressjs.com/en/4x/api.html>

npm install express



```
1  require('babel-register')
2  const express = require('express')
3  const app = express()
4
5  app.get('/api', (req,res) => {
6    res.send('Root API')
7  })
8
9  app.get('/api/v1', (req,res) => {
10   res.send('API Version 1')
11 })
12
13 app.listen(8080, () => console.log('Express OK (8080)'))
```

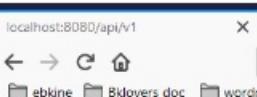
pour lancer l'app:

Donne:



localhost:8080/api

Root API



localhost:8080/api/v1

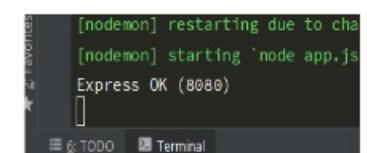
API Version 1



```
"license": "ISC",
"dependencies": {
  "babel-register": "^6.26.0",
  "express": "^4.16.4"
},
```

Pour démarrer expr.

1ère url



```
[nodemon] restarting due to change
[nodemon] starting `node app.js`
Express OK (8080)
```

2e url

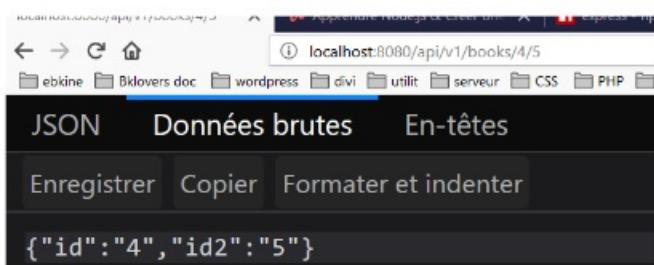
port à déterminer et msg

Cannot GET /qqch

Paramètres Url: utiliser le deux-points ':id' dans l'url

```
app.get('/api/v1/books/:id/:id2', (req, res) => {
  res.send(req.params)
})

app.listen(8080, () => console.log('Started on port 8080.'))
```



localhost:8080/api/v1/books/4/5

JSON Données brutes En-têtes

Enregistrer Copier Formater et indenter

{"id": "4", "id2": "5"}



express()
Application
Request
Properties
req.baseUrl
req.body
req.cookies
req.hostname
req.ip
req.ips
req.method
req.originalUrl
req.params
req.path

sur expressjs.com voir les propriétés dans les liens à gauche

☰ Définition des middlewares

Node.js formation app.js

Section 8, session 46

ex.:

```
app.use((req, res, next) => {
  console.log('URL : ' + req.url)
  next()
})
```

une fct ou module qui est lu quand il y a une requête quelle que soit la requête

app.use((req, res, next) =>{ ... next()})
ici, avant chaque app.get va s'exécuter un debug de l'url dans la console

☰ Utilisation du package Morgan

Section 8, session 47

D:\Développement\Node.js_formation (NodeJs_formation@1.0.0)
\$ npm install morgan |

morgan

npm v1.9.0 downloads 4M/month build passing coverage 100% gratipay no longer available

HTTP request logger middleware for node.js

sur npmjs.com

Named after **Dexter**, a show you should not watch until completion.

```
dev  
Concise output colored by response status for development use. The :status token will be colored red for server error codes, yellow for client error codes, cyan for redirection codes, and uncolored for all other codes.  
:method :url :status :response-time ms - :res[content-length]  
const express = require('express')  
const morgan = require('morgan')  
app.use(morgan('dev')) //dev est le mode que l'on veut mettre à morgan
```

```
Express OK (8080)  
GET /opopo 404 3.073 ms - 144  
GET /api 304 2.362 ms - -  
GET /api/v1/5/6 404 0.307 ms - 149  
GET /api/v1/books/5/6 200 0.659 ms - 20
```

☰ Gérer les paramètres en GET

Node.js formation app.js

```
const members = [  
  {  
    id: 1,  
    name: 'John'  
  },  
  {  
    id: 2,  
    name: 'Julie'  
  },  
  {  
    id: 3,  
    name: 'Bob'  
  }]
```

```
app.get('/api/v1/members/:id', (req, res) => {  
  res.send(members[(req.params.id)-1]) // -1 pour faire correspondre l'id à l'index
```

localhost:8080/api/v1/members/3
{"id":3,"name":"Bob"}

```
app.get('/api/v1/members/:id', (req, res) => {  
  res.send(members[(req.params.id)-1].name)
```

localhost:8080/api/v1/members/3
Bob

Param d'url

```
app.get('/api/v1/members', (req, res) => {
  if (req.query.max != undefined && req.query.max > 0) {
    res.send(members.slice(0, req.query.max))
  } else {
    res.send(members)
  }
})
```

```
[{"id":1,"name":"John"}, {"id":2,"name":"Julie"}]
```

49. Utilisation de Postman

API Dev environment

getpostman.com

Convention de réponses

Section 8, session 50

on va envoyer des réponses en json avec `res.json([body])`

```
p.get('/api/v1/members/:id', (req, res) =>
  res.json(func.success(member[(req.params.id)]))

p.get('/api/v1/members', (req, res) =>
  if (req.query.max != undefined && req.query.max > 0) {
    res.json(func.success(members.slice(0, req.query.max)))
  } else if (req.query.max != undefined && req.query.max < 0) {
    res.json(func.error('Wrong max value'))
  } else {
    res.json(func.success(members))
  }
)
```

Dans Postman

```
{
  "status": "error",
  "message": "Wrong max value"
}
```

création de functions.js dans node_module.

```
require('babel-register')
const func = require('functions')

exports.success = function(result) {
  return {
    status: 'success',
    result: result
  }
}

exports.error = function(message) {
  return {
    status: 'error',
    message: message
  }
}
```

création de fonctions de success et error intégrées aux réponses json

51. Gérer les paramètres en POST

il faut installer body-parser : `npm i body-parser`

`req.body`

Contains key-value pairs of data submitted in the request body. By default, it is `undefined`, and is populated when you use body-parsing middleware such as `body-parser` and `multer`.

The following example shows how to use body-parsing middleware to populate `req.body`.

```
var app = require('express')();
var bodyParser = require('body-parser');

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

app.use(bodyParser.json()); // for parsing application/json
app.use(bodyParser.urlencoded({ extended: true })) // for parsing application/x-www-form-urlencoded

app.post('/profile', upload.array(), function (req, res, next) {
  console.log(req.body);
  res.json(req.body);
})

app.post('/api/v1/members', (req, res) => {
  res.send(req.body)
})

app.listen(8080, () => console.log('Started'))
```

```

41 app.post('/api/v1/members', (req, res) => {
42
43     if (req.body.name) {
44
45         let member = {
46             id: members.length+1,
47             name: req.body.name
48     }
49
50         members.push(member)
51
52         res.json(func.success(member))
53
54     } else {
55         res.json(func.error('no name value'))
56     }
57 }

```

On teste avec Postman:

http://localhost:8080/api/v1/members

Key	Value	Description
name	Teddy	

```

1  {
2     "status": "success",
3     "result": [
4         {
5             "id": 1,
6             "name": "John"
7         },
8         {
9             "id": 2,
10            "name": "Julie"
11        },
12        {
13            "id": 3,
14            "name": "Bob"
15        },
16        {
17            "id": 4,
18            "name": "Teddy"
19        },
20        {
21            "id": 5,
22            "name": "Teddy"
23        }
24    ]
25 }

```

Développer un mécanisme pour ne pas envoyer 2 fois le même nom:

```

app.post('/api/v1/members', (req, res) => {

    if (req.body.name) {
        let sameName = false
        for (let i = 0; i < members.length; i++) {
            if (members[i].name === req.body.name){
                sameName = true
                break
            }
        }
        if (sameName) {
            res.json(func.error('Nom déjà pris'))
        } else {
            let member = {
                id: members.length+1,
                name: req.body.name
            }
            members.push(member)
            res.json(func.success(member))
        }
    } else {
        res.json(func.error('No name value'))
    }
})

```

vérification

.filter serait-il plus simple?

```

let valueToCheck = 'yep'

let arrayTest = ['yes', 'yep', 'okay']
console.log(arrayTest)

let filterArray = arrayTest.filter((test) => test === valueToCheck)
console.log(filterArray)

if (filterArray[0] != undefined) {
    console.log('In if.')
} else {
    console.log('In else.')
}

```

si sameName = erreur

sinon ajouter le membre

```

app.post('/api/v1/members', (req, res) => {

    if (req.body.name) {
        const sameName = members.filter((member) => member.name === req.body.name)
        if (sameName[0] != undefined){
            res.json(func.error('Nom déjà pris'))
        } else {
            let member = {
                id: members.length+1,
                name: req.body.name
            }
            members.push(member)
            res.json(func.success(member))
        }
    }
})

```

Ici, on développe la fonction post pour qu'elle ajoute un membre :id et nom à members ou un message d'erreur

var member id = members.length+1

push() ajoute le membre

réponse renvoie un success

Modifications du code

Node.js formation app.js

1- On peut transformer la variable func en objet {success, error} et retirer func dans le code.

```
const func = require('functions')
```

```
const {success, error} = require('functions')
```

2- Récuper un membre en trouvant l'index d'un id pour optimiser le code qd il y a des suppressions de membres dans l'index

création de la fonction getIndex

```
68
69  function getIndex(id){
70    for (let i = 0; i < members.length; i++) {
71      if (members[i].id == id)
72        return i
73    }
74    return 'Id invalide'
```

modification du app.get correspondant

```
21
22  app.get('/api/v1/members/:id', (req, res) => {
23
24    let index = getIndex(req.params.id);
25    if (typeof(index) === 'string') {
26      res.json(error(index))
27    } else {
28      res.json(success(members[index]))
29    }
30  })
31
32  res.json(success(members[(req.params.id)-1].name))
```

53. Gérer les requêtes PUT

```
39  app.put('/api/v1/members/:id', (req, res) => { //pour les modifications
40
41    let index = getIndex(req.params.id);
42
43    if (typeof(index) === 'string') {
44      res.json(error(index))
45    } else {
46      let same = false;
47      for (let i = 0; i < members.length; i++) {
48        if (req.body.name == members[i].name && req.params.id != members[i].id) {
49          same = true;
50          break;
51        }
52      }
53      if (same) {
54        res.json(error('Nom déjà utilisé'))
55      } else {
56        members[index].name = req.body.name;
57        res.json(success(true))
58      }
59    }
}
```

récup. l'index de l'id

si 'string' -> erreur

sinon: on vérifie si le nouveau nom est déjà utilisé dans un autre objet

oui -> erreur

non -> on change le nom

54. Gérer les requêtes DELETE

```
63  app.delete('/api/v1/members/:id', (req, res) => {
64
65    let index = getIndex(req.params.id);
66
67    if (typeof(index) === 'string') {
68      res.json(error(index))
69    } else {
70      members.splice(index, 1);
71      res.json(success(true))
72    }
73
74 })
```

création de createId()

```
115  function createId() {
116    return members[members.length - 1].id + 1
117  }
```

ajustement de la méthode post

```
92  } else {
93    let member = {
94      id: createId(),
95      name: req.body.name
96    }
97  }
98
99  res.json(member)
```

sinon répétition d'un même id, ici le id est créé après la récupération du dernier id de l'index +1

Test avec Postman toujours!

55. Créer des routes via Express

1- installation de router avec express.Router(): let MembersRouter = express.Router() au début

2- app.use('/api/v1/members), membersRouter) juste avant le listen

3- Un peu après le router, on peut installer un route: MembersRouter.route('/:id) et adapter le code associé:

- enlever app dans les fonctions
- on ne garde que le callback (enlève l'url)
- indente le code

```
19         name: 'Bob'  
20     }  
21 ]-  
22  
23     let MembersRouter = express.Router()  
24  
25     app.use(morgan('dev')) //dev est le mode que
```

```
    })-  
  
    app.use('/api/v1/members', MembersRouter)  
  
    app.listen(8080, () => console.log('Express OK (8080)'))  
  
function getIndex(id){
```

```
MembersRouter.route('/:id')  
  
.get((req, res) => {  
    let index = getIndex(req.params.id);  
  
.put((req, res) => { //pour les modifications  
    let index = getIndex(req.params.id);  
    if (typeof(index) === 'string') {  
  
.delete((req, res) => { //pour les suppressions  
    let index = getIndex(req.params.id);  
    if (typeof(index) === 'string') {
```

```
MembersRouter.route('/')  
  
.get((req, res) => {  
    if (req.query.max !== undefined && req.query.  
        max) {  
            res.send(`max=${req.query.max}`)  
        }  
    }  
  
.post((req, res) => {  
    if (req.body.name) {  
        const sameName = members.filter((member) =>  
            member.name === req.body.name);  
        if (sameName.length > 0) {  
            res.status(409).send(`Name ${req.body.name} already exists`);  
        } else {  
            const newMember = {  
                id: nanoid(),  
                name: req.body.name,  
                age: req.body.age  
            };  
            members.push(newMember);  
            res.status(201).send(`Member ${newMember.name} created`);  
        }  
    }  
});
```

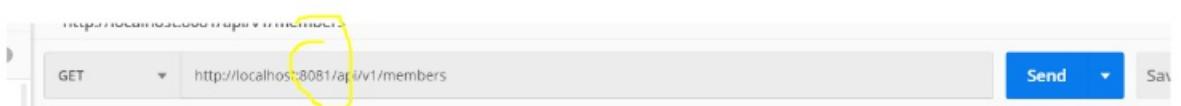
56. Création du config.json

```
1 {  
2     "rootAPI": "/api/v1/",  
3     "port": 8081  
4 }
```

Permet de modifier la route ou le port dans la config

```
5 const app = express()  
6 const morgan = require('morgan')  
7 const config = require('./config')
```

```
132     app.use(config.rootAPI+'members', MembersRouter)  
133  
134     app.listen(config.port, () => console.log('Started on port '+config.port))
```



Comprendre et gérer l'asynchrone sous Node.js

Section 9

58. Différence entre synchrone et asynchrone

Intéressant: Création d'un dir async d'un fichier app.js commande touch npm init --yes (package json) vite fait

```
cmd
D:\Développement\NodeJs_formation (NodeJs_formation@1.0.0)
λ mkdir async

D:\Développement\NodeJs_formation (NodeJs_formation@1.0.0)
λ cd async

D:\Développement\NodeJs_formation\async
λ touch app.js

D:\Développement\NodeJs_formation\async
λ npm init --yes
Wrote to D:\Développement\NodeJs_formation\async\package.json:

{
  "name": "async",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

D:\Développement\NodeJs_formation\async (async@1.0.0)
λ npm install babel-register
```

setTimeout crée l'asynchrone

Problème de la fonction asynchrone

```
require('babel-register')

console.log('Début')
getMember()
console.log('Fin')

function getMember() {
  setTimeout(() => {
    console.log('Member 1')
  }, 1500)
}
```

Nous verrons 3 patterns de ce problème

// Callbacks
// Promise
// Async / Await

59. Trois manières de gérer l'asynchrone

60. Les callbacks

Ne marche pas:

```
require('babel-register')

console.log('Début')
let member = getMember()
console.log(member)
console.log('Fin')

function getMember() {
  setTimeout(() => {
    return 'Member 1'
  }, 1500)
}
```

```
D:\Développement\NodeJs_formation\asy
λ node app.js
Début
undefined
Fin
|
node.exe
```

```
3   console.log('Début')
4   getMember((member) => {
5     console.log(member)
6     getArticles(member, (articles) => {
7       console.log(articles)
8     })
9   }
10  console.log('Fin')

12  function getMember(next) {
13    setTimeout(() => {
14      next('Member 1')
15    }, 1500)
16  }

18  function getArticles(member, next) {
19    setTimeout(() => {
20      next([1, 2, 3])
21    }, 1500)
22  }
```

ici, le code suivant fonctionne grâce à des fonctions de rappel

get member appelle member et affiche member quand il le reçoit

puis un coup fait il appelle l'array des articles avec une fonction de rappel aussi

L'ennui de cette syntaxe est

l'inclusion de callback qui devient moins lisible

on a donc inventé les:

«Promise»

next est une fonction spéciale pour gérer l'asynchrone, est mis à la place de return, c'est une sorte de convention fonctionnelle pour callback

certain appelle next : callback

début
fin



début
fin
member 1



```
[nodemon] starting node
début
fin
member 1
[ 1, 2, 3 ]
[nodemon] clean exit -
```

61. Les promesses

```
let p = new Promise((resolve, reject) => {  
})
```

`resolve` = fonction exécutée une fois que tout s'est bien fait dans la promesse ; `reject` si cela échoue

```
3  console.log('Début')  
4  let p = new Promise((resolve, reject) => {  
5      setTimeout(() => {  
6          resolve('All good.')  
7          //reject(new Error('Error during...'))  
8      }, 1500)  
9  })  
10  
11  p.then((message) => {  
12      console.log(message)  
13  }).catch((err) => {  
14      console.log(err.message)  
15  })  
16  
17  .then(gère si tout va bien et . catch si le processus n'a pas fonctionné)
```

Plus propre:

```
new Promise((resolve, reject) => {  
    setTimeout(() => {  
        resolve('Tout va bien!')  
        //reject(new Error('Problème rencontré...'))  
    }, 1500)  
}).then(message => console.log(message))  
.catch(err => console.log(err.message))  
console.log('fin')
```

62. Utilisation des promesses

```
require('babel-register')  
  
console.log('Début')  
  
getMember((member) => {  
    console.log(member)  
    getArticles(member, (articles) => {  
        console.log(articles)  
    })  
})  
  
function getMember(next) {  
    setTimeout(() => {  
        next('Member 1')  
    }, 1500)  
}  
  
function getArticles(member, next) {  
    setTimeout(() => {  
        next([1, 2, 3])  
    }, 1500)  
}  
  
// log('Fin')
```

```
require('babel-register')  
  
console.log('Début')  
  
getMember((member) => {  
    console.log(member)  
    getArticles(member, (articles) => {  
        console.log(articles)  
    })  
})  
  
function getMember() {  
    new Promise((resolve, reject) => {  
        setTimeout(() => {  
            resolve('Member 1')  
        }, 1500)  
    })  
}  
  
function getArticles(member, next) {  
    setTimeout(() => {  
        next([1, 2, 3])  
    }, 1500)  
}
```

```
require('babel-register')  
  
console.log('Début')  
  
getMember().  
    .then((member) => {  
        console.log(member)  
    })  
  
function getMember() {  
    return new Promise((resolve, reject) => {  
        setTimeout(() => {  
            resolve('Member 1')  
        }, 1500)  
    })  
}  
  
function getArticles(member, next) {  
    setTimeout(() => {  
        next([1, 2, 3])  
    }, 1500)  
}
```

```
getMember()  
    .then((member) => {  
        console.log(member)  
        getArticles(member)  
    })  
  
function getMember() {  
    return new Promise((resolve, reject) => {  
        setTimeout(() => {  
            resolve('Member 1')  
        }, 1500)  
    })  
}  
  
function getArticles(member) {  
    return new Promise((resolve, reject) => {  
        setTimeout(() => {  
            resolve([1, 2, 3])  
        }, 1500)  
    })  
}
```

```
console.log('Début')  
  
getMember()  
    .then(member => getArticles(member))  
    .then(articles => console.log(articles))  
  
function getMember() {  
    return new Promise((resolve, reject) => {  
        setTimeout(() => {  
            console.log('Member 1')  
            resolve('Member 1')  
        }, 1500)  
    })  
}  
  
function getArticles(member) {  
    return new Promise((resolve, reject) => {  
        setTimeout(() => {  
            resolve([1, 2, 3])  
        }, 1500)  
    })  
}
```

cmd
D:\Développement\Node.js_formation\à node app.js
Début
Fin
Member 1
[1, 2, 3]
D:\Développement\Node.js_formation\à

gestion des erreurs:

```
.then(member => getArticles(member))  
.then(articles => console.log(articles))  
.catch(err => console.log(err.message))
```

test erreur getMember

```
setTimeout(() => {  
    //console.log('Member 1')  
    //resolve('Member 1')  
    reject(new Error('Error during getMember()'))  
})
```

test erreur getArticles:

```
return new Promise((resolve, reject) => {  
    setTimeout(() => {  
        //resolve([1, 2, 3])  
        reject(new Error('Error during getArticles()'))  
    })  
})
```

Promesses en parallèle

Section 9, session 63

```

1  require('babel-register')
2
3  console.log('Début')
4
5  let p1 = new Promise((resolve, reject) => {
6      setTimeout(() => {
7          resolve('Promise 1')
8      }, 1500)
9  })
10
11 let p2 = new Promise((resolve, reject) => {
12     setTimeout(() => {
13         resolve('Promise 2')
14     }, 3000)
15 })
16
17 Promise.all([p1, p2])
18     .then(result => console.log(result))
19
20 console.log('Fin')

```

```

cmd
D:\Développement\NodeJs_formation\async (async@1.0.0)
λ node app.js
Début
Fin
[ 'p1', 'p2' ]

D:\Développement\NodeJs_formation\async (async@1.0.0)
λ node app.js
Début
Fin
[ 'Promise 1', 'Promise 2' ]

D:\Développement\NodeJs_formation\async (async@1.0.0)
λ node app.js
Début
Fin
[ 'Promise 1', 'Promise 2' ]

D:\Développement\NodeJs_formation\async (async@1.0.0)

```

.all: attend toutes les promesses avant d'afficher

.race: va donner la plus rapide

```

Promise.race([p1, p2])
    .then(result => console.log(result))

```

```

[ 'Promise 1', 'Promise 2' ]

D:\Développement\NodeJs_formation\async (async@1.0.0)
λ node app.js
Début
Fin
Promise 1

```

Async & Await autre syntaxe qui permet de faire des promesses

Section 9, session 64

ici viewArticles va attendre le resolve('Member1') de getMember()
et lancer à la suite le getArticles sans afficher le member 1

```

console.log('Début')

/*getMember()
    .then(member => getArticles(member))
    .then(articles => console.log(articles))*/
}

async function viewArticles() {
    let member = await getMember()
    let articles = await getArticles(member)
    console.log(articles)
}

viewArticles()

function getMember() {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            resolve('Member 1')
        }, 1500)
    })
}

function getArticles(member) {
    return new Promise((resolve, reject) => {

```

En utilisant cette syntaxe de fonction anonyme

```

    .then(articles => console.log(articles))*/
    () => {
        console.log('Milieu')
    }()
}

```

[attention ; devant cette anonyme; (point-virgule) seul cas obligatoire en js]

On aura plus simplement:

```

/*getMember()
    .then(member => getArticles(member))
    .then(articles => console.log(articles))*/
}

(async () => {
    let member = await getMember()
    let articles = await getArticles(member)
    console.log(articles)
})()

function getMember() {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            resolve('Member 1')
        }, 1500)
    })
}

function getArticles(member) {
    return new Promise((resolve, reject) => {

```

```

(async () => {
    try {
        let member = await getMember()
        let articles = await getArticles(member)
        console.log(articles)
    } catch (err) {
        console.log(err.message)
    }
})()

```

En gérant erreurs:

MySQL - Lier notre API REST à une base de données

Introduction de section Section 10, session 65

66. Mise en place de la base de données

Module MySQL - Connexion à la base de données

Node.js formation
Section 10, session 67

```
D:\Développement\NodeJs_formation (Node)
λ mkdir mysql-demo

D:\Développement\NodeJs_formation (Node)
λ cd mysql-demo\

D:\Développement\NodeJs_formation\mysql-de
λ touch app.js

D:\Développement\NodeJs_formation\mysql-de
λ npm init --yes
Wrote to D:\Développement\NodeJs_formation\

{
  "name": "mysql-demo",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\"",
    "keywords": [],
    "author": "",
    "license": "ISC"
  }
}

D:\Développement\NodeJs_formation\mysql-de
λ npm install babel-register mysql
cmd.exe
```

npmjs.com => mysql =>



Module MySQL - Fonctionnement des requêtes

Section 10, session 68

```
else {
  db.query('SELECT * FROM members', (err, result) => {
    if (err)
      console.log(err.message)
    else
      console.log(result)
  })
}
```

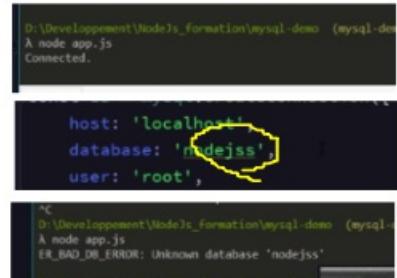
avec xampp phpmyadmin db=nodejs avec table
members
id int a_1 primary
name varchar 255

Establishing connections

The recommended way to establish a connection is this:

```
var mysql      = require('mysql');
var connection = mysql.createConnection({
  host     : 'example.org',
  user     : 'bob',
  password : 'secret'
});

connection.connect(function(err) {
  if (err)
    console.error('error connecting: ' + err.stack);
  else
    console.log('connected as id ' + connection.threadId);
});
```



```
^C
D:\Développement\NodeJs_formation\mysql-demo (mysql-de
λ node app.js
ER_BAD_DB_ERROR: Unknown database 'nodejs'
D:\Développement\NodeJs_formation\mysql-demo (mysql-de
```

```
db.query('SELECT * FROM members', (err, result) => {
  if (err)
    console.log(err.message)
  else
    console.log(result[0].name)
})
```

affichera "John"

affichera l'array

```
else {
  db.query('INSERT INTO members(name) VALUES("John")', (err, result) => {
    if (err)
      console.log(err.message)
    else
      console.log(result)
  })
}
```

insert John

```
Récupération des membres
Section 10, session 69
```

```
npm
D:\Développement\NodeJs_formation (NodeJs_format)
$ npm install mysql
```

```
const config = require('./config')

const db = mysql.createConnection({
  host: 'localhost',
  database: 'nodejs',
  user: 'root',
  password: ''
})

const members = [
  {
    id: 1,
    name: 'John Doe'
  },
  {
    id: 2,
    name: 'Jane Doe'
  }
]
```

On enlève members en dur

```
require('babel-register')
const [success, error] = require('bluebird')
const mysql = require('mysql')
const bodyParser = require('body-parser')
const express = require('express')
const app = express()
```

```
res.json(error('no name value'))

app.use(config.rootAPI+'members', MembersRouter)

app.listen(config.port, () => console.log(`Started on port ${config.port}`))

function getIndex(id) {
  const member = members.find(member => member.id === id)
  if (!member) return res.json(error('Member not found'))
  res.json(success(member))
}
```

saisi du code de l'app. sauf les fonctions

dans le else { }
de db.connect

```
const db = mysql.createConnection({
  host: 'localhost',
  database: 'nodejs',
  user: 'root',
  password: ''
})

db.connect((err) => {
  if (err)
    console.log(err.message)
  else {
    console.log('Connected.')
    let MembersRouter = express.Router()
    app.use(morgan('dev'))
    app.use(bodyParser.json())
    app.use(bodyParser.urlencoded({ extended: true }))
    MembersRouter.route('/:id')
      .get(getIndex)
      .put(updateMember)
      .patch(patchMember)
      .delete(deleteMember)
    app.use(config.rootAPI, MembersRouter)
  }
})
```

intégration de l'app sauf les fonctions dans le else de db.connect qu'on peut reprendre de mysql-demo et mettre sous les identifiants de connection

ajustement: déplacer const app

```
console.log('Connected.')

const app = express()
let MembersRouter = express.Router()

app.use(morgan('dev'))
```

```
const db = mysql.createConnection({
  host: 'localhost',
  database: 'nodejs',
  user: 'root',
  password: ''
})

db.connect((err) => {
  if (err)
    console.log(err.message)
  else {
    console.log('Connected.')
    let MembersRouter = express.Router()
    app.use(morgan('dev'))
    app.use(bodyParser.json())
    app.use(bodyParser.urlencoded({ extended: true }))
    MembersRouter.route('/:id')
      .get(getIndex)
      .put(updateMember)
      .patch(patchMember)
      .delete(deleteMember)
    app.use(config.rootAPI, MembersRouter)
  }
})
```

tester la connection.

```
// Récupère tous les membres
.get((req, res) => {
  if (req.query.max != undefined && req.query.max > 0)
    res.json(error(`Wrong max value`))
  else {
    res.json(success(members))
  }
})
```



On intègre les requêtes pour la bd:

```
// Récupère tous les membres
.get((req, res) => {
  if (req.query.max != undefined && req.query.max > 0) {

    db.query('SELECT * FROM members LIMIT 0, ? ', [req.query.max], (err, result) => {
      if (err) {
        res.json(error(err.message))
      } else {
        res.json(success(result))
      }
    })
  }
  else if (req.query.max != undefined) {
    res.json(error(`Wrong max value`))
  } else {

    db.query('SELECT * FROM members', (err, result) => {
      if (err) {
        res.json(error(err.message))
      } else {
        res.json(success(result))
      }
    })
  }
})
```

on teste

```

// Récupère un membre avec son ID
.get((req, res) => {

    let index = getIndex(req.params.id);

    if (typeof(index) == 'string') {
        res.json(error(index))
    } else {
        res.json(success(members[index]))
    }
})

// Récupère un membre avec son ID
.get((req, res) => {
    db.query('SELECT * FROM members WHERE id = ?', [req.params.id], (err, result) => {
        if (err) {
            res.json(error(err.message))
        } else {

            if (result[0] != undefined) {
                res.json(success(result))
            } else {
                res.json(error('Wrong id'))
            }
        }
    })
})

```

et on teste

ici requête protégée

on teste pour voir si on a un bon id
sinon erreur

70. Ajout d'un membre

```

// Ajoute un membre avec son nom
post((req, res) => {

    if (req.body.name) {

        db.query('SELECT * FROM members WHERE name = ?', [req.body.name], (err, result) => {
            if (err) {
                res.json(error(err.message))
            } else {

                if (result[0] != undefined) {
                    res.json(error('name already taken'))
                } else {

                    db.query('INSERT INTO members(name) VALUES(?)', [req.body.name], (err, result) => {
                        if (err) {
                            res.json(error(err.message))
                        } else {

                            db.query('SELECT * FROM members WHERE name = ?', [req.body.name], (err, result) => {
                                if (err) {
                                    res.json(error(err.message))
                                } else {

                                    res.json(success({
                                        id: result[0].id,
                                        name: result[0].name
                                    }))
                                }
                            })
                        }
                    })
                }
            }
        })
    }
})

```

Sera éventuellement organisé
avec des fonctions et classe
proprement

if (req.body.name) { C'était:

```

let sameName = false
for (let i = 0; i < members.length; i++) {
    if (members[i].name == req.body.name) {
        sameName = true
        break
    }
}

if (sameName) {
    res.json(error('name already taken'))
} else {
    let member = {
        id: createID(),
        name: req.body.name
    }
    members.push(member)
    res.json(success(member))
}

```

71. Modification d'un membre

```
.put((req, res) => { //pour les modifications
  if (req.body.name) {
    db.query('SELECT * FROM members WHERE id = ?', [req.params.id], (err, result) => {
      if (err) {
        res.json(error(err.message))
      } else {
        if (result[0] != undefined) {
          db.query('SELECT * FROM members WHERE name = ? AND id = ?', [req.body.name, req.params.id], (err, result) => {
            if (err) {
              res.json(error(err.message))
            } else {
              if (result[0] != undefined) {
                res.json(error('Nom déjà utilisé...'))
              } else {
                db.query('UPDATE members SET name = ? WHERE id = ?', [req.body.name, req.params.id], (err, result) => {
                  if (err) {
                    res.json(error(err.message))
                  } else {
                    res.json(success(true))
                  }
                })
              }
            }
          })
        }
      }
    })
  } else {
    res.json(error('Id invalide'))
  }
}
} else {
  res.json(error('Nom invalide (absent)'))
}
})
```

72 - Suppression d'un membre

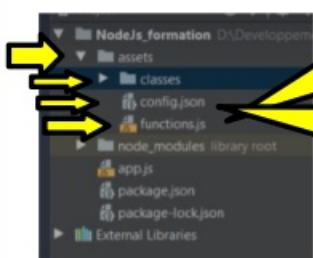
```
// Supprime un membre avec ID
.delete((req, res) => {

  db.query('SELECT * FROM members WHERE id = ?', [req.params.id], (err, result) => {
    if (err) {
      res.json(error(err.message))
    } else {
      if (result[0] != undefined) {
        db.query('DELETE FROM members WHERE id = ?', [req.params.id], (err, result) => {
          if (err) {
            res.json(error(err.message))
          } else {
            res.json(success(true))
          }
        })
      } else {
        res.json(error('Wrong id'))
      }
    }
  })
})
```

on peut enlever les fonctions indexOf et createId devenues inutiles

Création de la class Members

74. Restructuration des fichiers



```
require('babel-register')
const {success, error} = require('../assets/functions')
const mysql = require('mysql')
const bodyParser = require('body-parser')
const express = require('express')
const morgan = require('morgan')
const config = require('../assets/config')
```

app.js

```
{
  "rootAPI": "/api/v1/",
  "port": 8080,
  "db": {
    "host": "localhost",
    "database": "nodejs",
    "user": "root",
    "password": ""
}
```

config.json

```
const config = require('../assets/config')
const db = mysql.createConnection({
  host: config.db.host,
  database: config.db.database,
  user: config.db.user,
  password: config.db.password
})
```

app.js

Pour ajuster plus simplement la db dans un fichier de config

Tester avec Postman pour voir si tout va bien

75. Création et structure de la class

```
require('../assets/config')

.create
Name: members-class
Kind: JavaScript file
OK Cancel
```

dans assets/classes

```
let db, config
module.exports = Members
```

```
let db, config
module.exports = (_db, _config) => {
  db = _db
  config = _config
  return Members
}

let Members = class {
```

```
let MembersRouter = express.Router()
let Members = require('../assets/classes/members-class')
```

app.js

le require va passer db et config à member-class.js

```
let MembersRouter = express.Router()
let Members = require('../assets/classes/members-class')(db, config)
```

app.js

et Members est maintenant ds app.js

On peut:

```
app.use(morgan('dev')) // X
app.use(morgan())
6 const morgan = require('morgan')('dev')
```

test console:

```
let Members = require('../assets/classes/members-class')
console.log(Members)
```

```
[nodemon] starting 'node app.js'
Connected.
Started on port 8080
[nodemon] restarting due to changes...
[nodemon] starting 'node app.js'
Connected.
[Function: Members]
Started on port 8080
```

```
let Members = require('../assets/classes/members-class')
console.log(Members.getConfig())
```

```
let Members = class {
  static getConfig() {
    return config
  }
}
```

```
[nodemon] starting 'node app.js'
Connected.
{
  rootAPI: '/api/v1/',
  port: 8080,
  db: {
    host: 'localhost',
    database: 'nodejs',
    user: 'root',
    password: ''
  }
}
Started on port 8080
```

enlever ce console.log et getConfig! pour test s'il

76. Utilisation de "promise-mysql"

[promise-mysql](https://www.npmjs.com/package/promise-mysql)
3.3.1 • Public · [View on GitHub](#) · Published a month ago

[Readme](#) [4 Dependencies](#)

Promise-mysql

[Build passing](#)

Promise-mysql is a wrapper for [mysqljs/mysql](#) that wraps function
Usually this would be done with Bluebird's `.promisifyAll()` m
footprint is different to that of what Bluebird expects.

To install promise-mysql, use [npm](#):

```
$ npm install promise-mysql
```

<https://www.npmjs.com/package/promise-mysql>

To install promise-mysql, use [npm](#):

[\\$ npm install promise-mysql](#)

Connection

Important note: don't forget to call `connection.end()` when you're finished otherwise the Node process won't finish

To connect, you simply call `.createConnection()` like you would on [mysqljs/mysql](#):

```
var mysql = require('promise-mysql');
```

```
mysql.createConnection({
  host: 'localhost',
  user: 'sauron',
  password: 'theonetruering',
  database: 'mordor'
}).then(function(conn){
  // do stuff with conn
  conn.end();
})
```

```
password: config.db.password
}).then((db) => {
  // tout le code inclut ici
}).catch((err) => {
  console.log('Error during database connection')
  console.log(err.message)
})
```

terminal:

npm un mysql

npm i promise-mysql

app.js:

```
const {success, error} = require('./asset')
const mysql = require('promise-mysql')
const bodyParser = require('body-parser')

mysql.createConnection({
  host: config.db.host,
  database: config.db.database,
  user: config.db.user,
  password: config.db.password
})
```

```
password: config.db.password
}).then((db) => {
  // tout le code inclut ici
}).catch((err) => {
  console.log('Error during database connection')
  console.log(err.message)
})
```

```
console.log('Connected.')
const app = express()
let MembersRouter = express.Router()

members', MembersRouter)
() => console.log('Started on port '+config.port))]
```

enlever le reste:

```
)}
|
db.connect((err) => {
  if (err)
    console.log(err.message)
```

Récupération d'un membre - 1/2
Section 11, session 77

```
let Members = class {
  static getById()
```

static permet d'avoir directement la fonction sans avoir besoin d'instancier la classe

on pourra faire: Members.method

On récupère code de app.js du .get et on va l'adapter à promise-mysql

```
static getById(id) {  
    return new Promise((resolve, reject) => {  
        db.query('SELECT * FROM members WHERE id = ?', [id])  
            .then((result) => {  
                if (result.length > 0) {  
                    resolve(result[0]);  
                } else {  
                    reject(new Error('No member found with id: ' + id));  
                }  
            })  
            .catch((err) => reject(err))  
    });  
}
```

```
db.query('SELECT * FROM members WHERE id = ?', [req.params.id],  
    function (err, result) {  
        if (err) {  
            res.json(error(err.message)) // Généré par catch  
        } else {  
  
            if (result[0] === undefined) {  
                res.json(success(result[0]))  
            } else {  
                res.json(error('Wrong id'))  
            }  
        }  
    })
```

```
static getById(id) {  
  
    return new Promise((resolve, reject) => {  
  
        db.query('SELECT * FROM members WHERE id = ?', [id])  
            .then((result) => {  
                if (result[0] != undefined) {  
                    resolve(result[0])  
                } else {  
                    reject(new Error('Wrong id'))  
                }  
            })  
            .catch((err) => reject(err))  
  
    })  
}
```

Nous sommes dans **members-class**

Donc on renvoie l'array si ok

undefined pas de id
new Error

Si la connection se plante: err

app.js:

```
// Récupère un membre avec son ID
.get(async (req, res) => {
    let member = await Members.getByID(req.params.id)
    |
```

```
return new Promise((next) => {
    _____
    db.query('SELECT * FROM members WHERE
        .then((result) => {
            if (result[0] != undefined)
                next(result[0])
            else
                next(new Error('Wrong id')))
        })
        .catch((err) => next(err))
})
```

Pour gérer erreurs pas try/catch, mais:
(on enlève resolve et reject -> next)

```
// Récupère un membre avec son ID
.get(async (req, res) => {
    let member = await Members.get(
        req.params.id
    );
    if (member instanceof Error) {
        res.json(error(member.message));
    } else {
        res.json(success(member));
    }
});
```

réduire
tout
ça! 

78. Récupération d'un membre - 2/2

Pour cela, on va créer un fonction dans `fonctions.js`

```
exports.isErr = (err) => {
    return err instanceof Error;
}
```

app.js:

```
const {success, error, isErr} = require('../assets/functions')

.get(async (req, res) => {

    let member = await Members.getByID(req.params.id)
    res.json(isErr(member) ? error(member.message) : success(member))
})
```

On teste avec Postman

encore mieux:

```
  exports.checkAndChange = (obj) => {
    if (this.isErr(obj)) {
      return this.error(obj.message)
    } else {
      return this.success(obj)
    }
  }
```

```
const {success, error, checkAndChange} = require('../assets/functions')

.get(async (req, res) => {
    let member = await Members.getByID(req.params.id)
    res.json(checkAndChange(member))
})
```

On teste avec Postman

79. Récupération de tous les membres

```

// Récupère tous les membres
.get((req, res) => {
  if (req.query.max != undefined && req.query.max > 0) {
    db.query('SELECT * FROM members LIMIT 0, ?',
      [req.query.max]
    )
    .then((result) => res.json(result))
    .catch((err) => res.json({error: err.message}))
  } else {
    db.query('SELECT * FROM members')
    .then((result) => res.json(result))
    .catch((err) => res.json({error: err.message}))
  }
})
  
```

The code shows two ways to handle the 'max' query parameter. If it's defined and greater than 0, it performs a LIMIT 0, ? query. Otherwise, it performs a simple SELECT * query. Both paths result in JSON responses.

app.js est simplifié:

```

// Récupère tous les membres
.get(async (req, res) => {
  let allMembers = await Members.getAll(req.query.max)
  res.json(allMembers)
})
  
```

test

```

GET http://localhost:8080/api/v1/members?max=2
1: {
2:   "status": "error",
3:   "message": "ER_PARSE_ERROR: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '2' at line 1"
}
  
```

A screenshot of a browser showing a 500 Internal Server Error. The error message is: "ER_PARSE_ERROR: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '2' at line 1". This points to the line where the 'max' parameter is passed directly to the database query.

80. Ajout d'un membre

```

static add(name) {
  return new Promise((next) => {
    if (name) {
      db.query('SELECT * FROM members WHERE name = ?',
        [name]
      )
      .then((err) => {
        if (err) {
          next(new Error('no name value'))
        } else {
          db.query('INSERT INTO members(name) VALUES(?)',
            [name]
          )
          .then((err) => {
            if (err) {
              next(new Error('name already taken'))
            } else {
              db.query('SELECT * FROM members WHERE name = ?',
                [name]
              )
              .then((err, result) => {
                if (err) {
                  res.json({error: err.message})
                } else {
                  res.json({
                    id: result[0].id,
                    name: result[0].name
                  })
                }
              })
            }
          })
        }
      })
    }
  })
}
  
```

The code handles adding a member. It first checks if a name is provided. If yes, it performs a SELECT * query for that name. If no error, it performs an INSERT INTO query. Finally, it performs a SELECT * query again to get the newly added member's details and returns them in JSON.

```
//Crée un nouveau membre avec son nom
app.js
.post(async (req, res) => {
  let addMember = await Members.add(req.body.name)
  res.json(checkAndChange(addMember))
})
```

81. Modification d'un membre

```
// Modifie le nom d'un membre via son ID
static update(id, name) {
  return new Promise((next) => {
    if (name != undefined && name.trim() != '') {
      name = name.trim()

      db.query('SELECT * FROM members WHERE id = ?', [id])
        .then((result) => {
          if (result[0] != undefined) {
            return db.query('SELECT * FROM members WHERE name = ? AND id != ?', [name, id])
          } else {
            next(new Error(config.errors.wrongID))
          }
        })
        .then((result) => {
          if (result[0] != undefined) {
            next(new Error(config.errors.sameName))
          } else {
            return db.query('UPDATE members SET name = ? WHERE id = ?', [name, id])
          }
        })
        .then(() => next(true))
        .catch((err) => next(err))
    } else {
      next(new Error(config.errors.noNameValue))
    }
  })
}
```

- ▶ 82. Suppression d'un membre & Vue 04:08
globale

```
// Supprime un membre via son ID
static delete(id) {
  return new Promise((next) => {
    db.query('SELECT * FROM members WHERE id = ?', [id])
      .then((result) => {
        if (result[0] != undefined) {
          return db.query('DELETE FROM members WHERE id = ?', [id])
        } else {
          next(new Error(config.errors.wrongID))
        }
      })
      .then(() => next(true))
      .catch((err) => next(err))
  })
}
```

- ▶ 83. Messages d'erreurs dans config.errors

```
9  },
10 "errors": {
11   "wrongID": "It's a wrong id",
12   "wrongMaxValue": "wrong max value",
13   "nameAlreadyTaken": "name already taken",
14   "noNameValue": "no name value",
15   "sameName": "same name"
16 }
```

dans Members-class

```
  } else {
    next(new Error(config.errors.wrongID))
  }
})
.then((result) => {
  if (result[0] != undefined) {
    next(new Error(config.errors.sameName))
  } else {
```

etc.

Création de la documentation

Swagger & GitBook

doc externe au projet
doc interne au projet

Présentation de Swagger

Section 12, session 85

swagger.io -> Tools

suit les OpenAPI specification

from OpenAPI specification definitions

Swagger Editor

API editor for designing APIs with the OpenAPI Specification.

Swagger UI

Visualize OpenAPI Specification definitions in an interactive UI.

```
section 85 Swagger Editor File → Help → Generate Server → Generate Client →
1 swagger: "2.0"
2   info:
3     description: "this is a sample server Petstore server. You can find out more about
4       swagger at [http://swagger.io](http://swagger.io) or on [try.freemarker.net](http://try.freemarker.net). For this sample, you can use the api key 'special-key'
5       to test the authorization filters."
6     version: "1.0.0"
7     title: "Swagger Petstore"
8     termsOfService: "http://swagger.io/terms/"
9     contact:
10        email: "apiteam@swagger.io"
11     license:
12       name: "Apache 2.0"
13       url: "http://www.apache.org/licenses/LICENSE-2.0.html"
14     servers:
15       - url: "http://petstore.swagger.io"
16     tags:
17       - name: "pet"
18         description: "Everything about your Pets"
19         externalDocs:
20           description: "Find out more"
21           url: "http://swagger.io"
22       - name: "store"
23         description: "Operations about our store"
24         externalDocs:
25           description: "Find out more about our store"
26           url: "http://swagger.io"
27     schemes:
28       - "https"
29       - "http"
30       - "path"
31     /pet:
32       post:
33         summary: "Add a new pet to the store"
34         description: "Make it a pet"
35       /pet:
36         summary: "Add a new pet to the store"
37         description: "Make it a pet"
```

Swagger Petstore

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on try.freemarker.net/swagger/. For this sample, you can use the api key [special-key](#) to test the authorization filters.

Terms of service
Contact the maintainer
About 2.0
First time user about Swagger

Schemas

HTTPS

Authorize

pet Everything about your Pets
Find out more: <http://swagger.io>

store Access to Petstore orders
Find out more about our store: <http://swagger.io>

user Operations about user
Find out more about our store: <http://swagger.io>

A partir d'un fichier yaml ou json, swagger va produire une page complète html sur notre documentation

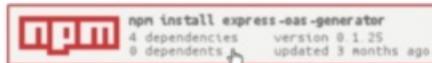
Swagger - Utilisation de express-oas-generator

Section 12, session 86 Nap Power: Maximum

Readme

4 Dependencies

express-oas-generator

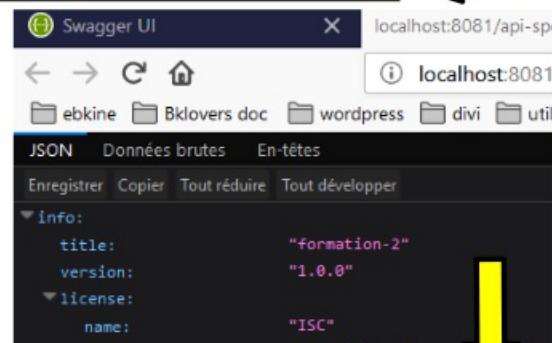
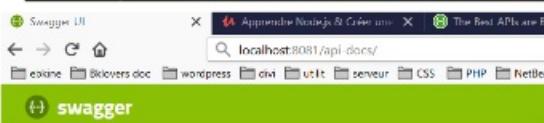


build passing coverage 93% vulnerabilities 0

```
C:\cmderr
λ cd "C:\Users\Don-de-Dieu\Desktop\node-formation 3"
$ Private packages, team management tools, and
C:\Users\Don-de-Dieu\Desktop\node-formation 3 (format
.0)
λ npm i express-oas-generator
npm WARN formation-2@1.0.0 No description
npm WARN formation-2@1.0.0 No repository field.
```

```
5 const express = require('express')_
6 const expressOasGenerator = require('express-oas-generator');

19 const app = express()
20 expressOasGenerator.init(app, {});
```



copier-coller

formation-2

Specification JSON



ISC

default



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Project node-formation 3 [node-formation 2]
  assets
    classes
      members-class
      config.json
      functions.js
      swagger.json
```

87. Swagger - Utilisation de swagger-ui-express

express-oas-generator
0.1.25 • Public • Published 3 months ago

Readme 4 Dependencies

Dependencies (4)

- express-list-endpoints
- generate-schema
- lodash
- swagger-ui-express

Swagger UI Express

Adds middleware to your express app to serve the Swagger UI bound to your Swagger document. This acts as living documentation for your API hosted from within your app.

Updated to Swagger 3.17.1

Usage

```

C:\Users\Don-de-Dieu\Desktop\node-formation 3 (formation-2@1.0.0)
$ npm i swagger-ui-express
npm WARN formation-2@1.0.0 No description
npm WARN formation-2@1.0.0 No repository field.

5  const express = require('express')
6  const swaggerUi = require('swagger-ui-express'); ←
7  const swaggerDocument = require('./assets/swagger.json'); ←
8  const morgan = require('morgan');

19  const app = express();
20  → expressOasGenerator.init(app); ←

29  app.use(bodyParser.urlencoded({ extended: true })); // for parsing application/x-www-form-urlencoded
30  → app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerDocument)); ←

29  app.use(bodyParser.urlencoded({ extended: true }));
30  → app.use(config.rootApi+'/api-docs', swaggerUi.serve); ←
    app.use(bodyParser.urlencoded({ extended: true }));
    app.use(config.rootApi+'api-docs');
  
```

localhost:8080/api/v1/api-docs/

formations-2

Specification JSON

ISC

default

GET /api/v1/members/{id} /api/v1/members/{id}

On peut préciser notre documentation dans ce fichier

88. Swagger - Comprendre le fichier swagger.json

Changer le titre

changer la descr.

ajouter un path et un host

enlever tous les paths qui était là: saisir le path **ctrl+maj+ D** et **delete**
pour les saisir tous et les supprimer

Préciser, modifier params, etc.

Préciser des tags avant "paths"

et dans chaque élément

members Méthodes permettant de gérer les utilisateurs

GET members/{id} members/{id}

default

members /members/{id} members/{id}

POST members

Parameters

Name max (query)

name * required (body)

Présentation de GitBook | <https://www.gitbook.com>

Section 12, session 89

GitBook

Get Started

- 1 Setup your organization
- 2 Create a project
- 3 Go live

ORGANIZATION URL
 Pick something short, memorable and recognizable for your organization's base address.

PROJECT'S TITLE
 This will be shown in the header of this project website.
It may differ from your organization's name.

Unlisted The content will only be visible to those with the link. It won't be indexed on Google.

NodeJs - API REST

Create a new release +

Accueil
Members
Récupération d'un membre
Récupération des membres
Untitled
+ Add New Page

Modification d'un membre

Last updated now

Page description (optional)

Enter your content here...

Or get started with a template:

Guide **API**

Request Response

200: OK Briefly describe this response example...

```
1 {
2   "status": "success",
3   "result": {
4     ... // Propriétés du membre
5   }
6 }
```

400: Bad Request Briefly describe this response example...

```
1 {
2   "status": "error",
3   "message": "wrong_id"
4 }
```

Récupération d'un membre

<https://localhost:8080/members/:id>

Briefly describe this method...

Request Response

Path Parameters

id	number	ID d'un membre
-----------	--------	----------------

+ Add a parameter

PUBLISHED modified 10 seconds ago

Next Acc

BP Describe what changed Bryan P. - 10 seconds ago

BP Describe what changed Bryan P. - 4 minutes ago

BP Initial revision Bryan P. - 11 minutes ago

End of history

PUBLISHED Bryan P. - 12 seconds ago

GitBook
Conclusion des deux outils https://...

Section 12, session 92

Test concret de l'API

Mise en place de l'application
Section 13, session 94

```

λ mkdir _front
D:\Développement\NodeJs_formation (NodeJs_formation@...
λ cd _front\
D:\Développement\NodeJs_formation\_front
λ touch app.js
D:\Développement\NodeJs_formation\_front
λ npm init --yes
Wrote to D:\Développement\NodeJs_formation\_front\package.json

{
  "name": "front",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

D:\Développement\NodeJs_formation\_front (_front@1.0.0)
λ npm install express body-parser babel-register
D:\Développement\NodeJs_formation\_front (_front@1.0.0)
λ npm install morgan --save-dev

```

_front/app.js

```

// Modules
require('babel-register')
const bodyParser = require('body-parser')
const express = require('express')
const morgan = require('morgan')('dev')

// Variables globales
const app = express()
const port = 8081

// Middlewares
app.use(morgan)
app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: true }))

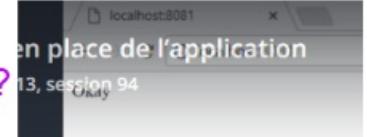
// Routes
app.get('/', (req, res) => {
  res.send('Okay')
})

// Lancement de l'application
app.listen(port, () => console.log('Started on port ' + port))

```

npm start

Tout baigne?



Langage de template - Twig
Section 13, session 95

pour chemin absolu.
dirname

test:

```

<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Title</title>
  </head>
  <body>
    <h1>Bonjour</h1>
  </body>
</html>

```

Bonjour

twig
1.12.0 • Public • P
npmjs.com Readme npm i twig https://twig.symfony.com = doc Symfony

index.html devient index.twig (le renommer)

```

<title>Title</title>
</head>
<body>
<h1>Bonjour {{ name }} ! </h1>
</body>
</html>

```

récupération des props par la vue

méthode render de twig:
param: (fichier de la vue, obj avec les propriétés)

```

// Routes
app.get('/', (req, res) => {
  res.render('index.twig', {
    name: 'Lucie'
  })
})

```

Title Modification Comparaison Coté serveur
localhost:8081/Francis
ebkine Bklovers doc wordpress divi util serveur

Bonjour Francis !

Amusons-nous!

space_name
className
_pipeName (Se
res.render('index.tw
name:_req.params.name|
})

96. Création du site web - Bootstrap

Home Documentation Examples Themes Expo Blog

Search...

Getting started
Introduction
Download
Contents
Browsers & devices
JavaScript
Theming
Build tools

Our `bootstrap.bundle.js` and `bootstrap.bundle.min.js` files include more information about what's included in Bootstrap.

Show components requiring JavaScript

Starter template

Be sure to have your pages set up with the latest conventions using an HTML5 doctype and including a viewport meta tag. Put it all together and your pages should look like this:

```
<!DOCTYPE html>
<html lang="en">
  <head>
```

Récuperer le starter de bootstrap pour index.twig

Search...

Getting started
Layout
Content
Reboot
Typography
Code
Images
Tables
Figures
Components

Striped rows

Use `.table-striped` to add zebra-striping to any table.

#	First	Last
1	Mark	Otto
2	Jacob	Thornt
3	Larry	the Bir

Et des tables...

```
<table class="table table-striped">
  ...
</table>
```

Petits ajustements de style

Page des membres - 1/2

Section 13, session 97

npm i axios

```
5 const morgan = require('morgan')('dev')
6 const axios = require('axios')
```

```
22   })
23 }
24
25 app.get('/members', (req, res) => {
26   axios.get('http://localhost:8081/api/v1/members')
27     .then((response) => {
28       console.log(response)
29     })
30     .catch((err) => renderError(res, err.message))
31 })
32 }
```

Fonction pour gérer le catch et page 404

```
36 //Fonctions
37 = error.twig
38
39 function renderError(res, errMsg) {
40   res.render('error.twig', {
41     errMsg: errMsg
42   })
43 }
```

Création de error.twig

```
views
  error.twig
```

```
<title>Erreur</title>
</head>
<body class="p-5">
  <h4 class="mb-5 text-center">{{ errMsg }}</h4>

  <!-- Optional JavaScript -->
  <!-- jQuery first, then Popper.js, then Bootstrap.js -->
```

On utilisera Axios pour les requêtes HTTP

axios

0.18.0 • Public • Published 5 months ago

Readme 2 Dependencies

axios

Promise based HTTP client for the browser and node.js

Features



On débogue: on voit si on reçoit une réponse visible dans Cmder

```
_onInactiveResponse: [Function],
_currentRequest: [Circular],
_currentUrl: 'http://localhost:8081/api/',
[Symbol(isWeakRef)]: false,
[Symbol(headersKey)]: { accept: [Array],
[Array], host: [Array] } },
data:
{ status: 'success',
result:
[ [Object], [Object], [Object], [Object],
[Object] ] }
```

On copie index et on ajuste

Création de members.twig:

```

views
  error.twig
  index.twig
  members.twig

```

```

8   <!-- Bootstrap CSS -->
9   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
10  <title>Membres</title>
11  </head>
12  <body class="p-5">
13    <h1 class="mb-5 text-center">Tous les membres</h1>
14    <table class="table table-striped">
15      <thead>

```

```

[nodemon] restarting due to change
[nodemon] starting 'node app.js'
Démarré sur le port 8080
rs
[nodemon] starting 'node app.js'
Démarré sur le port 8080
[ { id: 2, name: 'Julia1' },
  { id: 3, name: 'Robert' },
  { id: 4, name: 'Francis' },
  { id: 7, name: 'Jules2' },
  { id: 8, name: 'Irène' },
  { id: 9, name: 'Renée' } ]
GET /members - - ms -

```

Création d'une instance de requêtes

You can create a new instance of axios with a custom configuration object.

```

//variables globales
const app = express();
const port = 8080;
const fetch = axios.create({
  baseURL: 'http://localhost:8081/api/v1',
  timeout: 1000,
  headers: { 'X-Custom-Header': 'foobar' }
});

```

autre debug

```

app.get('/members', (req, res) => {
  axios.get('http://localhost:8081/api/v1/members')
    .then((response) => {
      if (response.data.status === 'success') {
        console.log(response.data.result)
      } else {
        renderError(res, response.data.message)
      }
    })
    .catch((err) => renderError(res, err.message))
})

```

on a renommé 'instance': fetch

```

//variables globales
const app = express();
const port = 8080;
const fetch = axios.create({
  baseURL: 'http://localhost:8081/api/v1',
  timeout: 1000,
  headers: { 'X-Custom-Header': 'foobar' }
});

```

98. Page des membres - 2/2

```

app.get('/members', (req, res) => {
  fetch.get('/members')
    .then((response) => {
      if (response.data.status === 'success') {
        if success: res.render('members.twig', {
          members: response.data.result
        })
      } else {
        renderError(res, response.data.message)
      }
    })
})

```

genre de foreach/php en twig : { % (logique) % }

Dans l'en-tête, on récupère

```


| id | name  | grade |
|----|-------|-------|
| 1  | John  | free  |
| 3  | Julie | free  |
| 4  | Lucas | free  |
| 5  | Yan   | free  |


```

dans le corps: les {{value}} de member pour chacun des member in members

```


| id | name  | grade |
|----|-------|-------|
| 1  | John  | free  |
| 3  | Julie | free  |
| 4  | Lucas | free  |
| 5  | Yan   | free  |


```

Tous les membres

id	name	grade
1	John	free
3	Julie	free
4	Lucas	free
5	Yan	free

autre test: stopper mysql

Apache	9212	443, 8090	<button>Stop</button>	<button>Addr</button>
MySQL	12732	3308	<button>Stop</button>	<button>Addr</button>

Cannot enqueue Query after fatal error.

99. Page d'un membre & Changements

```

app.get('/members/:id', (req, res) => {
  fetch.get('/members/' + req.params.id)
    .then((response) => {
      if (response.data.status === 'success') {
        res.render('member.twig', {
          member: response.data.result
        })
      } else {
        renderError(res, response.data.message)
      }
    })
    .catch((err) => renderError(res, err.message))
})

```

copie ajustée de members.twig

```

// Page d'accueil
app.get('/', (req, res) => {
  // Page récupérant tous les membres
  app.get('/members', (req, res) => {
    <thead>
    <tr>
      <% for index, value in member %>
        <th scope="col">{{ index }}</th>
      <% endfor %>
    </tr>
    </thead>
    <tbody>
      <tr>
        <% for value in member %>
          <td>{{ value }}</td>
        <% endfor %>
      </tr>
    </tbody>
  })
})

```

Optimisation du code:

```
function apiCall(url, res, next) {
    fetch.get(url)
        .then((response) => {
            if (response.data.status == 'success') {
                next(response.data.result)
            } else {
                renderError(res, response.data.message)
            }
        })
        .catch((err) => renderError(res, err.message))
}
```

Création de apiCall()

```
/member/:id
// Page récupérant un membre en fonction de son ID
app.get('/members/:id', (req, res) => {
    apiCall('/members/'+req.params.id, res, (result) => {
        res.render('member.twig', {
            member: result
        })
    })
})
```

so, we call it!

Ajustement /members

```
// Page récupérant tous les membres
app.get('/members', (req, res) => {
    apiCall('/members', res, (members) => {
        res.render('members.twig', {
            members: members
        })
    })
})
```

so, we call it!


```
// Page récupérant tous les membres
app.get('/members', (req, res) => {
    apiCall('/members', res, (result) => {
        res.render('members.twig', {
            members: result
        })
    })
})
```

changement de members par result

avec max= `apiCall(req.query.max ? '/members?max='+req.query.max : '/members', res, (result) => { res.render('members.twig', {` une petite ternaire!

Page de modifications

File View Node.js Section 13, session 100

views edit.twig

<title>Modification</title>

enter" > Modification</h1>

edit.twig

```
<h1 class="mb-5 text-center">Modification</h1>
<form>
    {% for index, value in member %}
        <div class="form-group">
            <label for="{{ index }}>{{ index }}</label>
            <input type="text" class="form-control" id="{{ index }}" name="{{ index }}" value="{{ value }}>
        </div>
    {% endfor %}
    <button type="submit" class="btn btn-primary">Modifier</button>
</form>
```

Pour que le id ne soit pas modifiable:

un if sur une vue twig:
`value="{{ value }}" {% if index == 'id' %} readonly {% endif %}`

```
48 app.get('/edit/:id', (req, res) => {
49     apiCall('/members/'+req.params.id, res, (result) => {
50         res.render('edit.twig', {
51             member: result
52         })
53     })
54 })
```

Ceci permet d'afficher la page pour éditer

activons le «form»

<form action="" method="post">

Il faut gérer un post: envoie les données à l'api rest

Modif de apiCall pour gérer les différentes méthodes: get, post, etc.

```
function apiCall(url, method, data, res, next) {
    fetch({
        method: method,
        url: url,
        data: data
    }).then((response) => {
        if (response.data.status == 'success') {
            next(response.data.result)
        } else {
            renderError(res, response.data.message)
        }
    })
    .catch((err) => renderError(res, err.message))
}
```

ajuster les 3 app.get déjà écrites

```
s?max='+req.query.max : '/members', 'get', {}, res, (result) => {
    apiCall('/members/'+req.params.id, 'get', {}, res, (result) => {
        res.render('edit.twig', {
            member: result
        })
    })
})
```

Méthode post redirige sur '/members' et on en profite pour faire pareil avec '/'

```
app.post('/edit/:id', (req, res) => {
    apiCall('/members/'+req.params.id, 'put', {
        name: req.body.name
    }, res, () => {
        res.redirect('/members')
    })
})
```

// Page d'accueil

```
app.get('/', (req, res) => {
    res.redirect('/members')
```

Testons!

Donc on peut supprimer index.html

101. Méthode de suppression

Créer un bouton sur edit.twig pour supprimer un membre:

```
</div>
  {% endfor %}
  <button type="submit" class="btn btn-primary">Modifier</button>
</form>

<form action="/delete" method="post" class="mt-5"> le input est caché
  <input type="hidden" name="id" value="{{ member.id }}>
  <button type="submit" class="btn btn-primary">Supprimer</button>
</form>
```

John

Modifier

Supprimer

de suppression

session 101

Cannot POST /delete

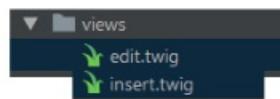
app.js

```
60   res.redirect('/members')
61 }
62 })
63
64 app.post('/delete', (req, res) => {
65   apiCall('/members/' + req.body.id, 'delete', {}, res, () => {
66     res.redirect('/members') On redirige vers members
67   })
68 })
```

Commentons: 64 // Méthode permettant de supprimer un membre

102. Page d'ajout d'un membre

// Page |
72 app.get('/insert', (req, res) => {
 res.render('insert.twig')
}) Affichage



<title>Créer un membre</title>
<head>
><h1>Création d'un membre</h1>

```
<form action="" method="post">

  <div class="form-group">
    <label for="name">Nom du nouveau membre</label>
    <input type="text" class="form-control" id="name" name="name">
  </div>

  <button type="submit" class="btn btn-primary">Ajouter</button>
</form>
```

76 //méthode pour créer le membre
77 app.post('/insert', (req, res) => {
78 apiCall('/members/', 'post', {
79 name: req.body.name
80 }, res, () => {
81 res.redirect('/members')
82 })
83 }) traitement du formulaire

Nom du nouveau membre

Bertrand

10

Benoit

Ajouter

12

Bertrand

Liens divers & Conclusion

Section 13, session 103

Pour faire des liens sur members:

```
</thead>
<tbody>
  {% for member in members %}
    <tr style="cursor: pointer;" onclick="document.location='/edit/{{ member.id }}'">
      {% for value in member %}
```

32 </tbody>
33 </table>
34 Ajouter un membre



Enfin, on peut faire des variantes!

```
<a href="/edit/{{ member.id }}" class="btn btn-primary mt-2">Modifier ou supprimer ce membre</a><br>
<a href="/" class="btn btn-primary mt-2">Accueil</a>
```

```
insert.twig
25
26 <a href="/" class="btn btn-primary mt-2">Accueil</a>
27
```

```
edit.twig
<a href="/" class="btn btn-primary mt-2">Accueil</a>
```

```
error.twig
<h4 class="mb-5 text-center">{{ errorMsg }}</h4>
<a href="/members" class="btn btn-primary">Retour à la liste des membres</a>
```

Fin!