Notes de cours :

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!

Section : 3 **Comprendre Node.js**

Environnement exécutant du Javascript sans navigateur et open-source.

Développement d'application back-end (API)

Entreprises

- → Microsoft
- → LinkedIn
- → PayPal

Points forts

- → Applications en temps réel
- → Applications scalables
- → Javascript (langage front-end)
- → Grosse communauté
- → Très nombreuses librairies (modules)

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

Chaque navigateur a son propre moteur JS

Ryan Dahl

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

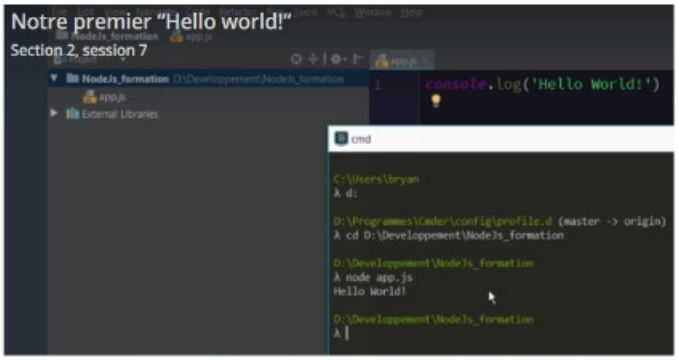
Fonctionnement interne
Section 3, session 11

Javascript est conçu pour gérer les évènements, 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.

Commande bloquante	Commande non-bloquante
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

Notre premier "Hello world!"
Section 2, session 7



installer nodemon:

Nodemon éxécute automatiquement le code dans terminal

nodemon app.js pour démarrer

rs en console pour rafraichir

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

Utilisation d'ECMAScript6 (ES6)

Section 4

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

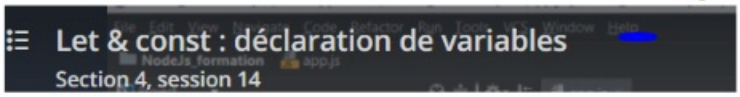
Babel est maintenant inutile avec node.js

app.js

```
1   require('babel-register');
2
3   console.log('test')
```

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

scope dans les blocs ({ entre accolades })



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
14              fs.readFile('test.txt', 'utf-8', (err, data) => {
15                  console.log(data)
16              })
17          })
18      }
19  })
```

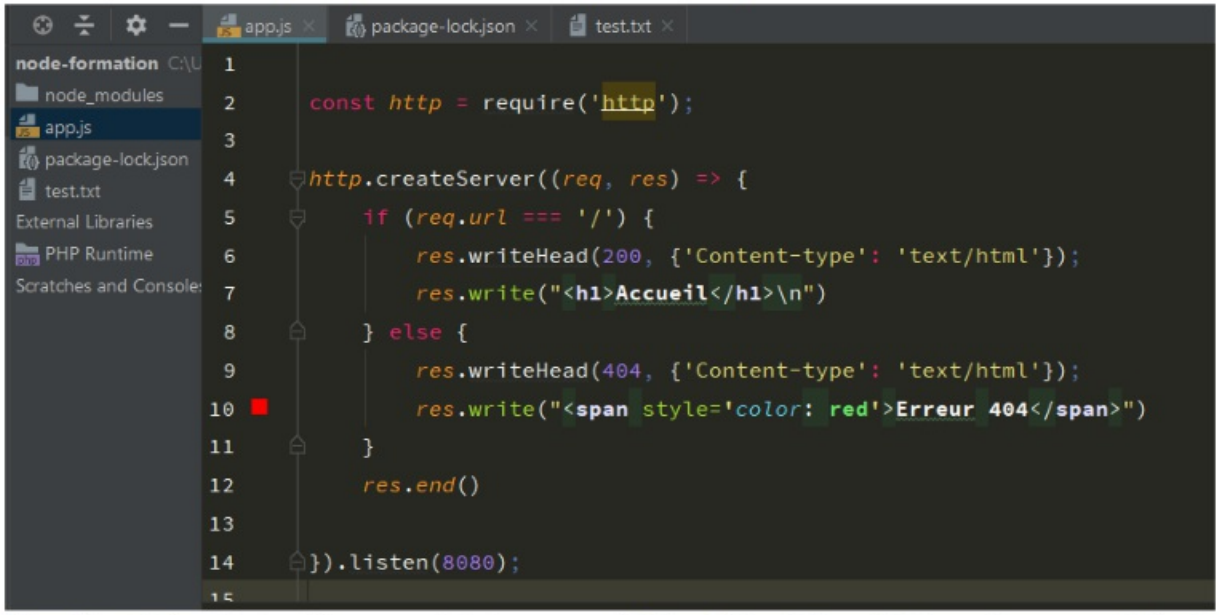
// x64

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

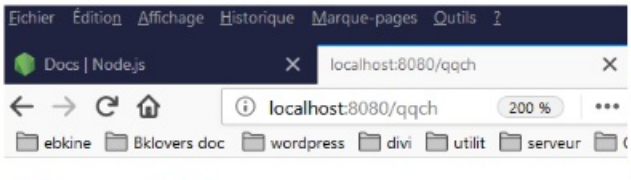
// lit le fichier (fs.readFile())

// écrit dans le fichier fs.writeFile()

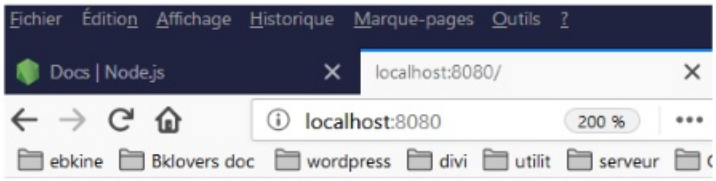
Module http: req =requête res=response http.createServer res.write res.writeHead



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

http.createServer((req, res) => {
    if (req.url === '/') {
        res.writeHead(200, {'Content-type': 'text/html'});
        res.write("<h1>Accueil</h1>\n")
    } else {
        res.writeHead(404, {'Content-type': 'text/html'});
        res.write("<span style='color: red'>Erreur 404</span>")
    }
    res.end()

}).listen(8080);
```



Erreur 404



Accueil

23 - Combinaison de modules

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

http.createServer((req, res) => {
    if (req.url === '/') {
        res.writeHead(200, {'Content-type': 'text/html'});
        res.write("<h1>Accueil</h1>\n")
        res.end()
    } else if (req.url === '/test') {
        fs.readFile('test.txt', 'utf-8', (err, data) => {
            if (err) {
                send404(res);
            } else {
                res.writeHead(200, {'Content-type': 'text/html'});
                res.write(data);
                res.end()
            }
        })
    } else {
        send404(res);
    }
}).listen(8080);


function send404(res){
    res.writeHead(404, {'Content-type': 'text/html'});
    res.write("<span style='color: red'>Erreur 404</span>")
    res.end()
}
```

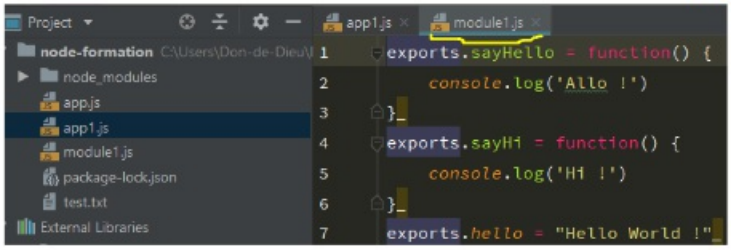
Ici, on a deux if router: /test renvoie le contenu de test.txt

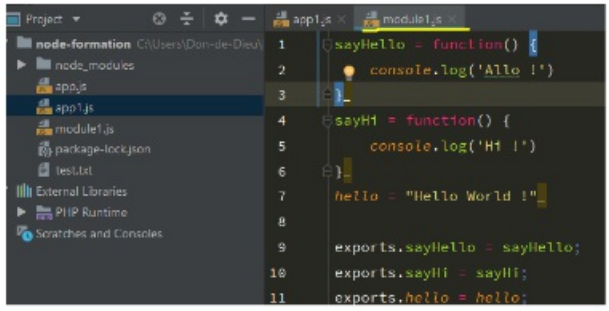
res.end a dû être déplacé et doit être répété, car par exemple le readFile envoie une réponse asynchrone, ce qui fait planter l'app si on ferme la réponse dans le champ global.

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

24-faire ses modules

ou





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

package.json: **npm init**
npm init --yes (val. par défaut)

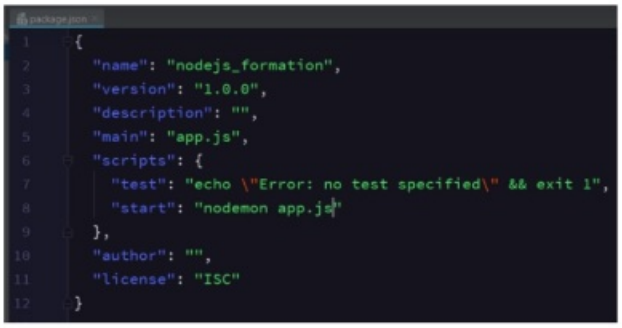
NPM

→ npmjs.org

→ Node Package Manager

→ Command line

→ « apt-get » de Linux



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.

<u>Installation de module avec npm</u>

ex. avec babel-register: sur npmjs.com, on aura les instructions d'installation, "dépendancies" 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

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:\Developpement\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: 6ed021173e2fcb486d7acb45c6009a856f647071

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 <bng412@gmail.com>
- thejameskyle <me@thejameskyle.com>
- sebmck <sebmck@gmail.com>
- danez <daniel@tschinder.de>
- hzoo <hi@henryzoo.com>
- loganfsmyth <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:\Developpement\NodeJs_formation  (nodejs_formation@1.0.0)
λ
```

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

```
D:\Developpement\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:\Developpement\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:\Developpement\NodeJs_formation  (nodejs_formation@1.0.0)
λ
```

Installer une version particulière:

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

Mise à jour:

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

babel-register    6.1.4  6.26.0  6.26.0  nodejs_formation

D:\Developpement\NodeJs_formation  (nodejs_formation@1.0.0)
λ
```

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:\Developpement\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
```

⇨

```
  },
  "devDependencies": {
    "ms": "^2.1.1"
  }
```

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

⇨

```
D:\Developpement\NodeJs_formation  (nodejs_formation@1.0.0)
λ node app.js
172800000

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

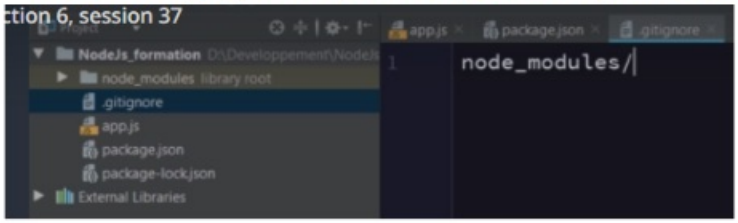
36. Comprendre le modules globaux (-g)

```
D:\Developpement\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

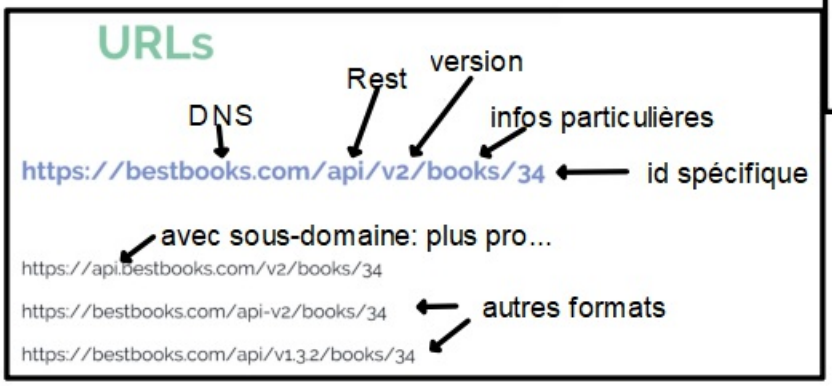
- → Representational state transfer
- → Architecture de requêtes i.e. de liaisons entre client et serveur
- → RESTful Pour parler de cette architecture
- → API

Requêtes (CRUD)

- → Create
- → Read
- → Update
- → Delete

Commandes normales ou courantes faites à une base de donnée

URLs



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

un objet

```
{
  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 - Création d'une API REST

Installation de express qui va gérer les url
voir doc: http://expressjs.com/en/4x/api.html
npm install express

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

app.js | .gitignore | package.json

```
require('babel-register')
const express = require('express')
const app = express()

app.get('/api', (req,res) => {
    res.send('Root API')
})
app.get('/api/v1', (req,res) => {
    res.send('API Version 1')
})

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

(plus de res.end ou res.write ni de if, express simplifie la syntaxe avec **app.get**)

Pour démarrer expr.

1ère url

2e url

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

TODO | Terminal

pour lancer l'app:

port à déterminer et msg

Donne:

localhost:8080/api — Root API

localhost:8080/api/v1 — API Version 1

localhost:8080/qqch — 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"}
```

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

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

Définition des middlewares
Section 8, session 46

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

ex.:

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

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:\Developpement\NodeJs_formation (NodeJs_formation@1.0.0)
λ 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]
```

```
3  const morgan = require('morgan')
```

```
6  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
```

6: TODO | Terminal

Gérer les paramètres en GET
Section 8, session 48

```
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'idex
})_
```

localhost:8080/api/v1/members/3

ebkine | Bklovers doc | wordpress | divi | utilit | serveur | CSS

```
{"id":3,"name":"Bob"}
```

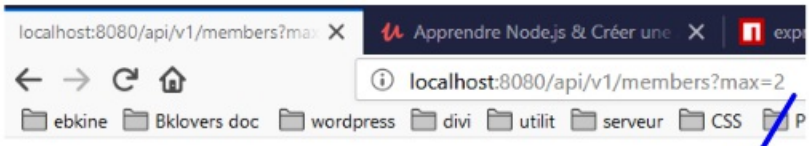
```
app.get('/api/v1/members/:id', (req, res) => {
    res.send(members[(req.params.id)-1].name)
})_
```

localhost:8080/api/v1/members/3

ebkine | Bklovers doc | wordpress | divi | utilit | serveur | CSS

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, re
  res.json(func.success(member[(req.p

p.get('/api/v1/members', (req, res) =
  if (req.query.max != undefined && r
      res.json(func.success(members.s
  } else if (req.query.max != undefin
      res.json(func.error('Wrong max
  } else {
      res.json(func.success(members))
```

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

création de functions.js dans node_module.

création de fonctions de success et error intégrées aux réponses json

app.js | functions.js | .gitignore | package.jso

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

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

Dans Postman

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

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()); // for parsing application/json
app.use(bodyParser.urlencoded({ extended: true })); // for parsi

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

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

```
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
55      } else {
56          res.json(func.error('no name value'))
57      }
```

Ici, on développe la fonction post pour qu'elle ajoute un membre :id et nom

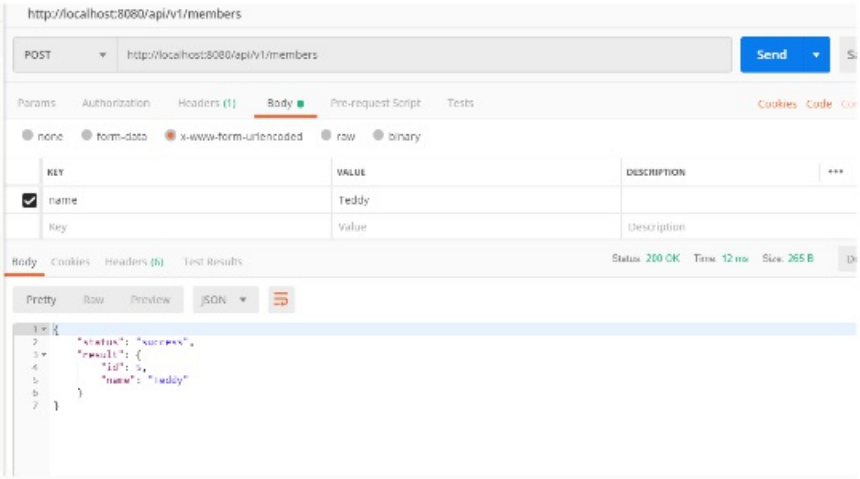
à members ou un message d'erreur

var member id = members.length+1

push() ajoute le membre

réponse renvoie un success

On teste avec Postman:



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))
```

Modifications du code
Section 8, session 52

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

```
28  app.get('/api/v1/members/:id', (req, res) => {
29
30      let index = getIndex(req.params.id);
31      if (typeof(index) === 'string') {
32          res.json(error(index))
33      } else {
34          res.json(success(members[index]))
35      }
36
37  })
```

```
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

```
53  app.delete('/api/v1/members/:id', (req, res) => {
54
55      let index = getIndex(req.params.id);
56
57      if (typeof(index) === 'string') {
58          res.json(error(index))
59      } 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              }
```

sinon répétion 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 functions**
- **on ne garde que le callback (enlève l'url)**
- **indente le code**

```
        name: 'Bob'
    }
]

let MembersRouter = express.Router()

app.use(morgan('dev')) //dev est le mode que
```

```
})

app.use('/api/v1/members', MembersRouter)

app.listen(8080, () =>  console.log('Express OK (808

function getIndex(id){
```

```
MembersRouter.route('/:id')

    .get((req, res) => {
        let index = getIndex(req.params.id);
```

```
MembersRouter.route('/')

    .get((req, res) => {
        if (req.query.max !== undefined && req.query.
```

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

```
    .post((req, res) => {
        if (req.body.name) {
            const sameName = members.filter((member) =
```

```
    .delete((req, res) => {  //pour les suppressions
        let index = getIndex(req.params.id);
```

56. Création du config.json

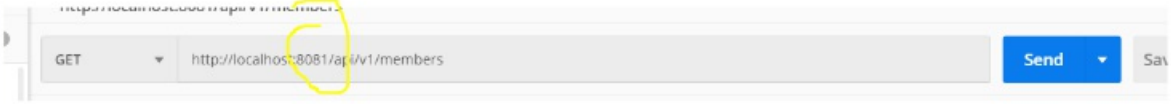
```
const morgan = require('morgan')
const config = require('./config')
```

app.js | config.json | functions.js

```
{
  "rootAPI": "/api/v1/",
  "port": 8081
}
```

Permet de modifier la route ou le port dans la config

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

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



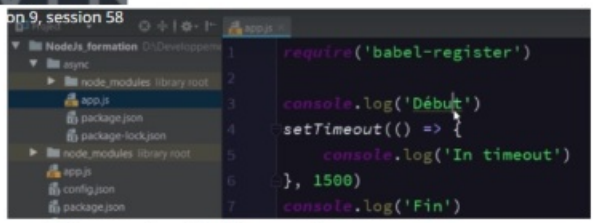
Comprendre et gérer l'asynchrone sous Node.js

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:\Developpement\NodeJs_formation  (NodeJs_formation@1.0.0)
λ mkdir async

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

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

D:\Developpement\NodeJs_formation\async
λ npm init --yes
Wrote to D:\Developpement\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:\Developpement\NodeJs_formation\async  (async@1.0.0)
λ npm install babel-register
```



setTimeout crée l'asynchrone

Problème de la function 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:\Developpement\NodeJs_formation\asy
λ node app.js
Début
undefined
Fin
node.exe
```

```
console.log('Début')
getMember((member) => {
    console.log(member)
    getArticles(member, (articles) => {
        console.log(articles)
    })
})
console.log('Fin')

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

function getArticles(member, next) {
    setTimeout(() => {
        next([1, 2, 3])
    }, 1500)
}
```

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
```

⇨

```
début
fin
member 1
[ 1, 2, 3 ]
[nodemon] clean exit -
6: TODO  Terminal
```

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

```
console.log('Début')
let p = new Promise((resolve, reject) => {

    setTimeout(() => {
        resolve('All good.')
        //reject(new Error('Error during...'))
    }, 1500)

})

p.then((message) => {
    console.log(message)
}).catch((err) => {
    console.log(err.message)
})
```

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')
```

.then gère si tout va bien et . catch si le processus n'a pas fonctionné

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)
}
```

```
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)
    })
```

```
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:\Developpement\NodeJs_formation\a
λ node app.js
Début
Fin
Member 1
[ 1, 2, 3 ]

D:\Developpement\NodeJs_formation\a
```

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:

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

Promesses en parallèle
Section 9, session 63

```
require('babel-register')

console.log('Début')

let p1 = new Promise((resolve, reject) => {
    setTimeout(() => {
        resolve('Promise 1')
    }, 1500)
})

let p2 = new Promise((resolve, reject) => {
    setTimeout(() => {
        resolve('Promise 2')
    }, 3000)
})

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

console.log('Fin')
```

```
cmd

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

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

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

D:\Developpement\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(resu
```

```
[ 'Promise 1', 'Promise 2' ]

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

Async & Await
Section 9, session 64

autre syntaxe qui permet de faire des promesses

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]

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) => {
```

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) => {
```

En gérant erreurs:

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

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

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

Module MySQL - Connexion à la base de données
Section 10, session 67

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

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

D:\Developpement\NodeJs_formation\mysql-d
λ touch app.js

D:\Developpement\NodeJs_formation\mysql-d
λ npm init --yes
Wrote to D:\Developpement\NodeJs_formatio

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

D:\Developpement\NodeJs_formation\mysql-d
λ npm install babel-register mysql
cmd.exe
```

npmjs.com => mysql =>

mysql

2.16.8 • Public • Published 6 months ago

Readme 4 Depe

mysql

npm v2.16.0 downloads 1M/m node >= 0.6 linux passing

Table

- Install
- Introc

```
require('babel-register')
const mysql = require('mysql')

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

db.connect((err) => {

    if (err)
        console.log(err.message)
    else
        console.log('Connected.')

})
```

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);
    return;
  }

  console.log('connected as id ' + connection.threadId);
});
```

```
D:\Developpement\NodeJs_formation\mysql-demo  (mysql-de
λ node app.js
Connected.
```

```
    host: 'localhost',
    database: 'nodejss',
    user: 'root',
```

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

Module MySQL - Fonctionnement des requêtes
Section 10, session 68

```
else {

    console.log('Connected.')

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

}
```

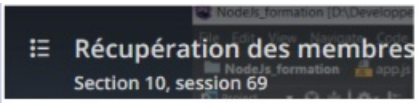
affichera l'array

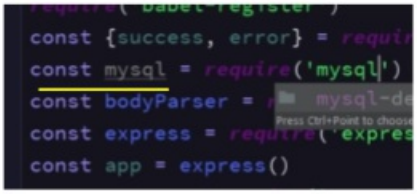
```
db.query('SELECT * FROM members', (err, result) => {
    if (err)
        console.log(err.message)
    else
        console.log(result[0].name)
})
```

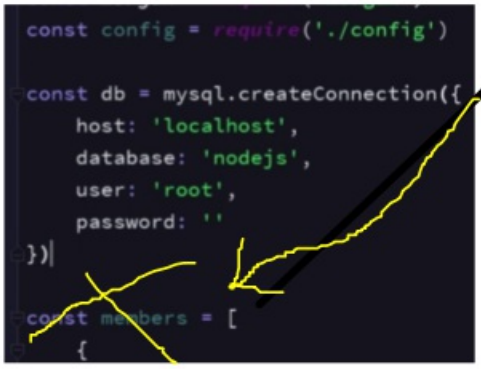
affichera "John"

```
else {

    console.log('Connected.')

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

}
```

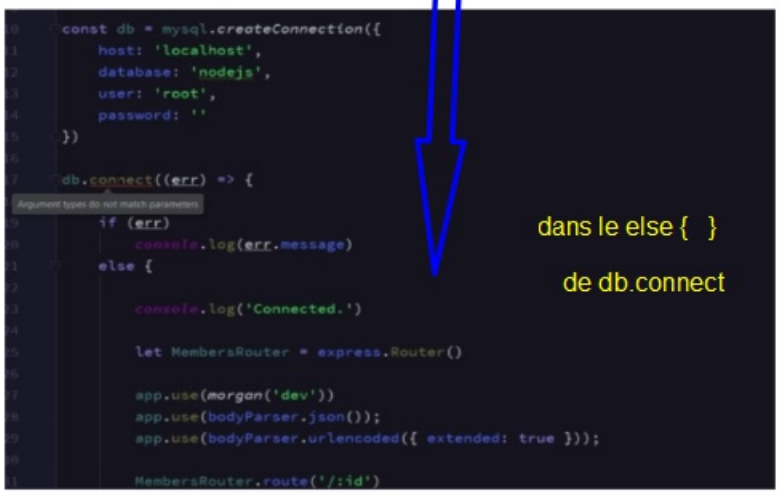
insert John





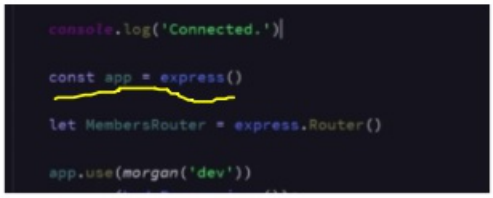


On enlève members en dur

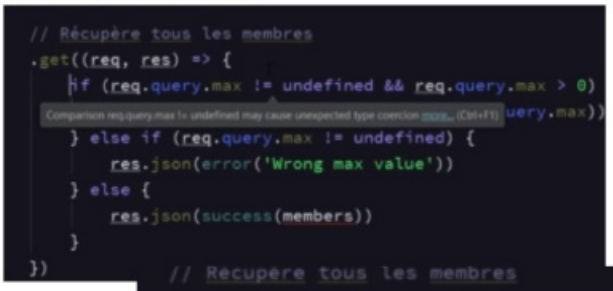


intégration de l'app sauf les fonctions dans le else de db.connect qu'on peut reprendre de mysql-demo et mettre sous les identifiants de connection

ajustement: déplacer const app



tester la connection.



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]))
    }

})
```

et on teste

```
// 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'))
            }

        }
    })
```

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

C'était:

```
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(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)'))
    }
})
```

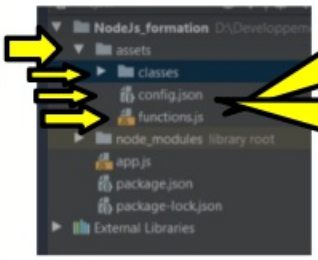
72 - Suppression d'un membre

```
// 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')
nst 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/conf

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

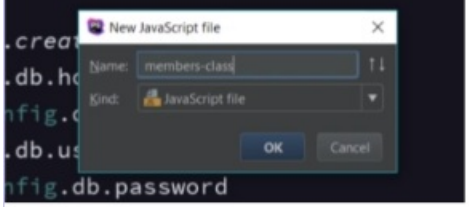
app.js

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

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

.creat
.db.ho
nfig.c
.db.us
nfig.db.password
```



dans assets/classes

```
let db, config

module.exports = Members

let Members = class {

}
```

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

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

app.js

```
let db, config

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

let Members = class {

}
```

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

app.js

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

et Members est maintenant ds app.js

On peut:

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

```
app.use(morgan)
```

```
const morgan = require('morgan')('dev')
```

test console:

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

```
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/cl
console.log(Members.getConfig())
```

```
let Members = class {

    static getConfig() {

        return config

    }
}
```

```
[nodemon] starting `node
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 slt

76. Utilisation de "promise-mysql"

promise-mysql

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

Readme | 4 Dependencies

Promise-mysql

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
```

va simplifier le code

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();
});
```

terminal:

npm un mysql

npm i promise-mysql

app.js:

```
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) => {

            })
            .catch((err) => reject(err))

    })
```

```
db.query('SELECT * FROM members WHERE id = ?', [req.params
    if (err) {
        res.json(error(err.message))
    } else {

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

géré par catch

```
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)

    |
```

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

```
return new Promise((next) => {

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

dans class Members

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

    let member = await Members.getByID(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

app.js

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

Member-Class

```
static getAll(max) {

    return new Promise((next) => {

        if (max != undefined && max > 0) {

            db.query('SELECT * FROM members LIMIT 0, ?',
                if (err) {
                    res.json(error(err.message))
                } else {
                    res.json(success(result))
                }
```

```
    return new Promise((next) => {

        if (max != undefined && max > 0) {
            db.query('SELECT * FROM members LIMIT 0, ?', [max])
                .then((result) => next(result))
                .catch((err) => next(err))
        } else if (max != undefined) {
            next(new Error('Wrong max value'))
        } else {
            db.query('SELECT * FROM members')
                .then((result) => next(result))
                .catch((err) => next(err))

        }
    })
}
```

app.js est simplifié:

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

test

GET http://localhost:8080/api/v1/members?max=2

```
{
    "status": "error",
    "message": "ER_PARSE_ERROR: You have an error in your SQL syn
        for the right syntax to use near ''2'' at line 1"
```

```
static getAll(max) {

    return new Promise((next) => {

        if (max != undefined && max > 0) {
            db.query('SELECT * FROM members LIMIT 0, ?', [parseInt(max)])
```

80. Ajout d'un membre

```
static add(name) {

    return new Promise((next) => {

        if (name) {

            db.query('SELECT * FROM me
                if (err) {
```

```
return new Promise((next) => {

    if (name != undefined && name.trim() != '') {

        name = name.trim()
```

```
db.query('SELECT * FROM members WHERE name = ?', [name])
    .then((result) => {

        if (result[0] != undefined) {
            next(new Error('name already taken'))
        } else {

            db.query('INSERT INTO members(name) VALUES(?
                if (err) {
```

...

```
} else {
    next(new Error('no name value'))
}
```

```
        next(new Error('name already taken'))
    } else {
        return db.query('INSERT INTO members(name) VALUES(?)', [name])
    }

})
.then(() => {
    db.query('SELECT * FROM members WHERE name = ?', [name], (err, result) => {

        if (err) {
            res.json(error(err.message))
        } else {
            res.json(success({
                id: result[0].id,
                name: result[0].name
            }))
        }
```

```
})
.then(() => {
    return db.query('SELECT * FROM members WHERE name = ?', [name])
})
.then((result) => {
    next({
        id: result[0].id,
        name: result[0].name
    })
})
.catch((err) => next(err))
```

app.js

```
//Crée un nouveau membre avec son nom
.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 globale 04:08

```
// 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

config.json

```
  },
  "errors" : {
    "wrongID": "It's a wrong id",
    "wrongMaxValue": "wrong max value",
    "nameAlreadyTaken": "name already taken",
    "noNameValue": "no name value",
    "sameName": "same name"
  }
}
```

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 interne au projet

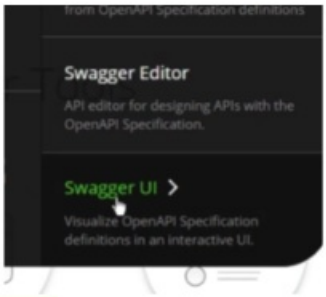
doc externe au projet

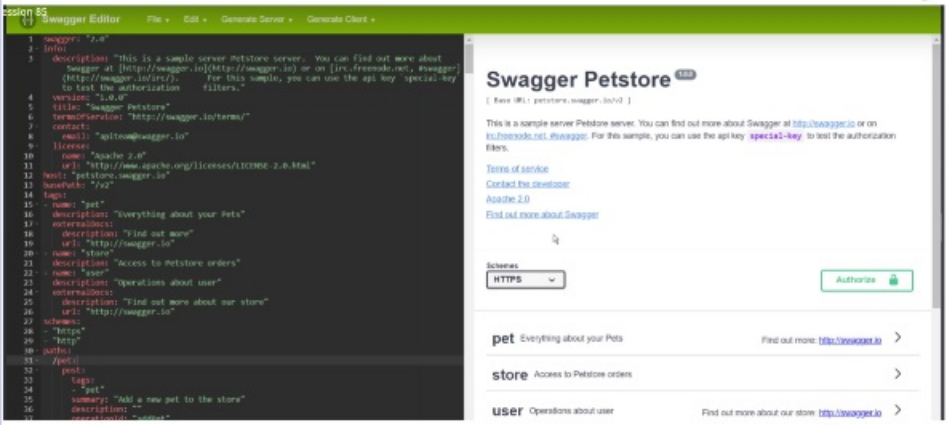
Présentation de Swagger

Section 12, session 85

swagger.io -> Tools

suit les OpenAPI specification

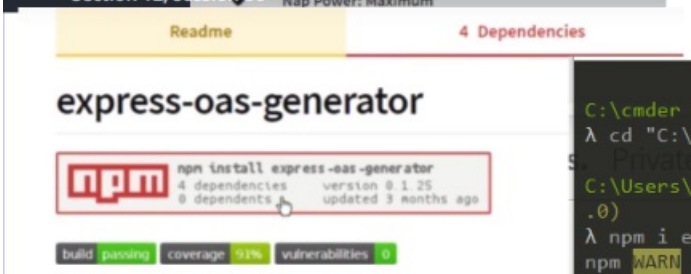




A partir d'un fichier yaml ou json, swagger va produire
une page complète html sur notre documentation

Swagger - Utilisation de express-oas-generator

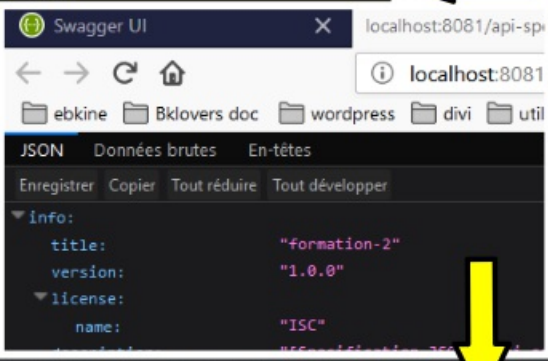
Section 12, session 86



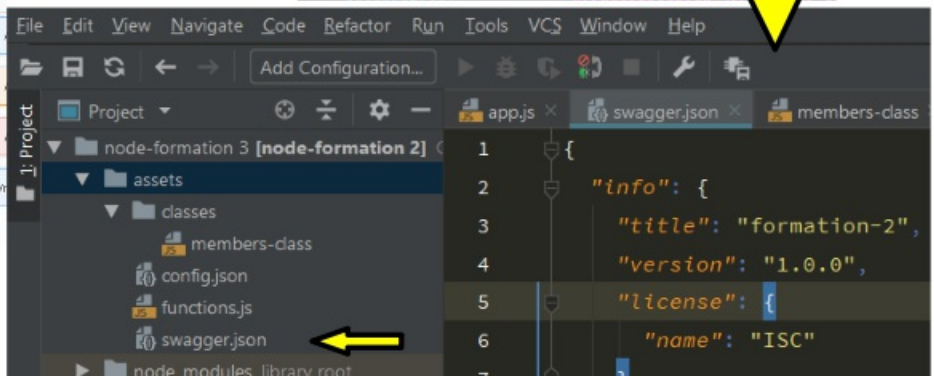

```
C:\cmder
λ cd "C:\Users\Don-de-Dieu\Desktop\node-formation 3"
C:\Users\Don-de-Dieu\Desktop\node-formation 3  (format
.0)
λ npm i express-oas-generator
npm WARN formation-2@1.0.0 No description
npm WARN formation-2@1.0.0 No repository field.
```

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

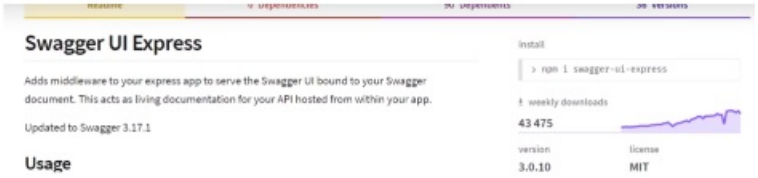
```
19      const app = express()
20      expressOasGenerator.init(app, {});
```



copier-coller



87. Swagger - Utilisation de swagger-ui-express




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

```
λ npm un express-oas-generator
npm WARN formation-2@1.0.0 No description
npm WARN formation-2@1.0.0 No repository field.
```

```
const express = require('express')
const swaggerUi = require('swagger-ui-express');
const swaggerDocument = require('./assets/swagger.json');
```

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

```
app.use(bodyParser.urlencoded({ extended: true })); // for parsing applic
app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerDocument));
```

```
app.use(bodyParser.urlencoded({ extended: tr
app.use(config.rootApi+'/api-docs', swaggerU
```

```
app.use(bodyParser.urlencoded({ e
app.use(config.rootApi+'api-docs'
```

88. Swagger - Comprendre le fichier swagger.json

On peut préciser notre documentation dans ce fichier

```
{
  "info": {
    "title": "API REST documentation",
    "version": "1.0.0",
    "license": {
      "name": "ISC"
    },
    "description": "API REST pour gérer des utilisateurs"
  },
  "basePath": "/api/v1",
  "host": "localhost:",
  "paths": {
    "members/{id}": {
```

Changer le titre

changer la descr.

ajouter un path et un host

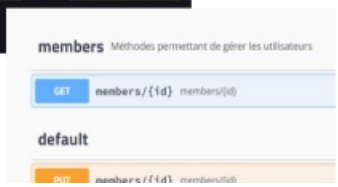
enlever tous les paths qui était là: saisir le path ctrl+maj+ D et delete pour les saisir tous et les supprimer


```
"tags": [
  {
    "name": "members",
    "description": "Méthodes permettant de gérer les utilisateurs"
  }
],
```

préciser des tags avant "paths"

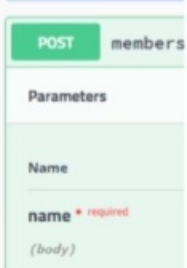
et dans chaque élément

```
"members/{id}": {
  "get": {
    "tags": ["members"],
    "summary": "members/{id}"
```



Préciser, modifier params, etc.

```
],
"parameters": [
  {
    "name": "name",
    "in": "body",
    "required": true
  }
]
```



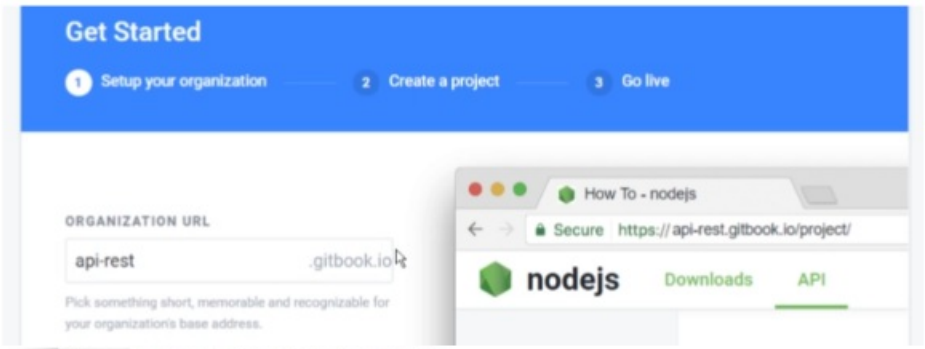
Présentation de GitBook

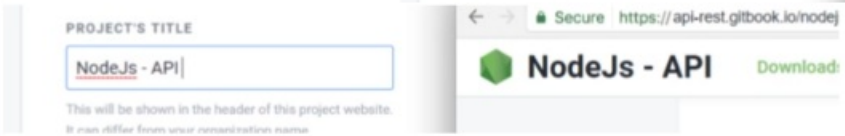
Section 12, session 89

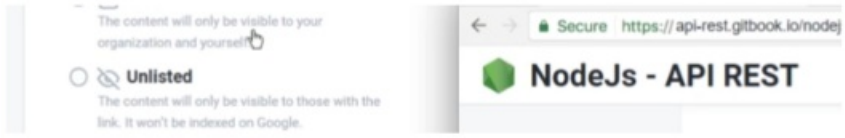
https://www.gitbook.com

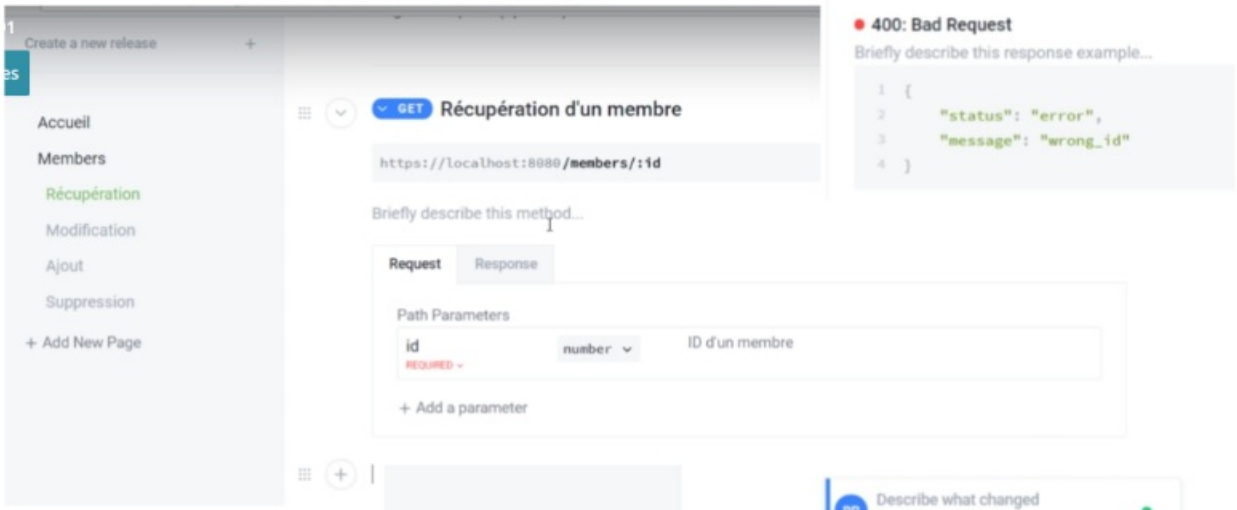
GitBook

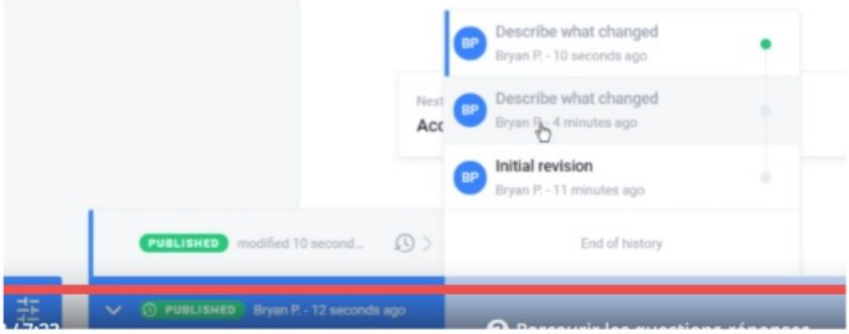
s'inscrire











Conclusion des deux outils
Section 12, session 92

Test concret de l'API

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

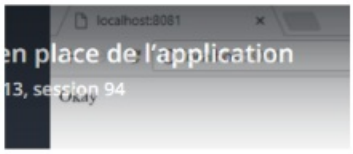
```
λ mkdir _front

D:\Developpement\NodeJs_formation  (NodeJs_formation@
λ cd _front\

D:\Developpement\NodeJs_formation\_front
λ touch app.js

D:\Developpement\NodeJs_formation\_front
λ npm init --yes
Wrote to D:\Developpement\NodeJs_formation\_front\pac

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


D:\Developpement\NodeJs_formation\_front  (_front@1.0
λ npm install express body-parser b abel-register
```

```
D:\Developpement\NodeJs_formation\_front  (_front@1.0
λ npm install morgan --save-dev
```

```
"test": "echo \"Error: no test
"start": "nodemon app.js"
```

npm start

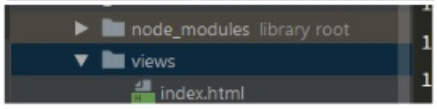
Tout baigne?




```
// 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))
```

_front/app.js

Langage de template - Twig
Section 13, session 95

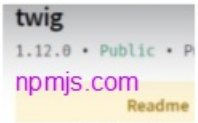



```
app.get('/', (req, res) => {
    res.sendFile(__dirname +'/views/index.html')
```

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>
```



npm i twig

= doc Symfony

```
const morgan = require('morgan')('dev')
const twig = require('twig')
```

app.js

```
    <title>Title</title>
</head>
<body>

<h1>Bonjour {{ name }} ! </h1>

</body>
</html>
```

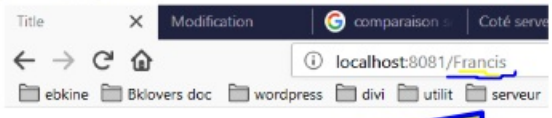
index.html devient index.twig (le renommer)

récupération des props par la vue

méthode render de twig:

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

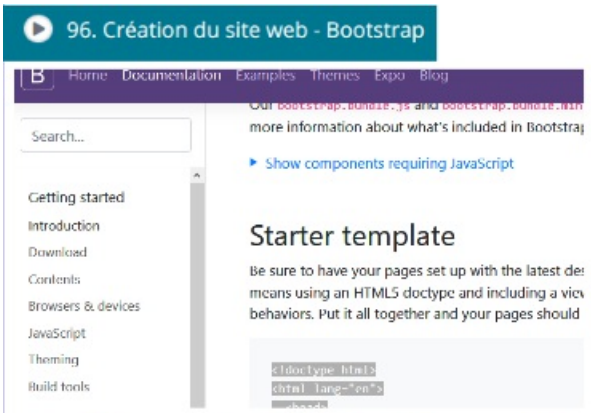
param: (fichier de la vue, obj avec les propriétés)

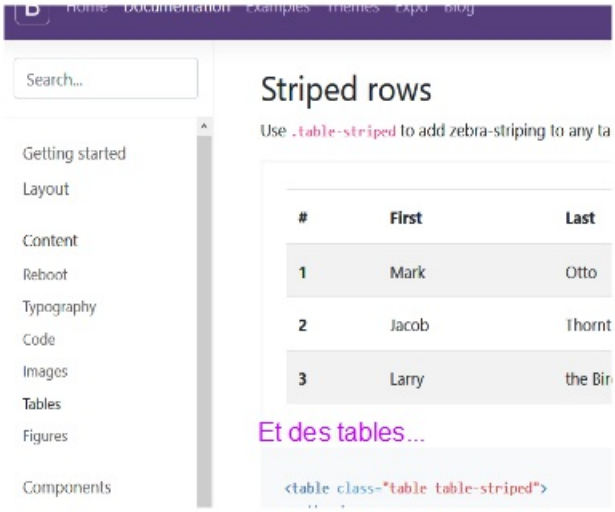


Bonjour Francis !

Amusons-nous!

```
// Routes
app.get('/:name', (req,
    res.render('index.tw
        name: req.params.name
    })
})
```






```
</head>
<body class="p-5"> = padding: 5px
<h1 class="mb-5 text-center">Page</h1>  = margin-bottom: 5px; text-align:center;

<table class="table table-striped">
    <thead>
    <tr>
        <th scope="col">#</th>
```

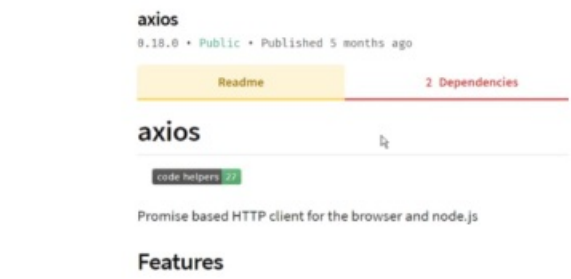
Petits ajustements de style

Page des membres - 1/2
Section 13, session 97

On utilisera Axios pour les requêtes HTTP

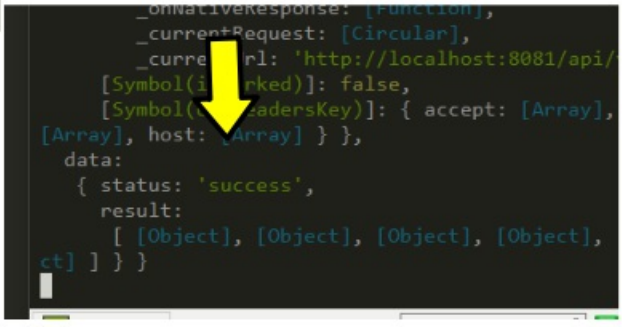
npm i axios

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




```
    })
})

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

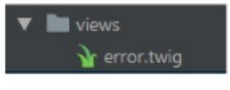
On débogue: on voit si on reçoit une réponse: visible dans Cmder




```
//Fonctions                    = error.twig

function renderError(res, errMsg) {
    res.render('error.twig', {
        errorMsg: errMsg
    })
}
```

Fonction pour gérer le catch et page 404

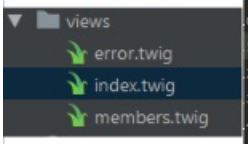
Création de error.twig




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

<!-- Optional JavaScript -->
```

On copie index et on ajuste

Création de members.twig:




```
<!-- Bootstrap CSS -->
<link rel="stylesheet" href="https://stackpath.bo
<title>Membres</title>
</head>
<body class="p-5">
<h1 class="mb-5 text-center">Tous les membres</h1>
<table class="table table-striped">
<thead>
```

autre débug

```
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))
```

```
[nodemon] restarting due to cha
[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: 'Irènee' },
  { id: 9, name: 'Renée' } ]
GET /members - - ms - -
```

node.exe

Création d'une instance de requêtes

https://www.npmjs.com/package/axios

wordpress | divi | utilit | serveur | CSS | PHP

You can create a new instance of axios with a custom conf

axios.create([config])

```
var instance = axios.create({
  baseURL: 'https://some-domain.com/api/',
  timeout: 1000,
  headers: {'X-Custom-Header': 'foobar'}
});
```

on a renommé 'instance': fetch

```
//variables globales
const app = express()
const port = 8080
const fetch = axios.create({
    baseURL: 'http://localhost:8081/api/v1',
})
```

```
app.get('/members', (req, res) => {
    fetch.get('/members')
        .then((response) => {
```

98. Page des membres - 2/2

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

if success:

else

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

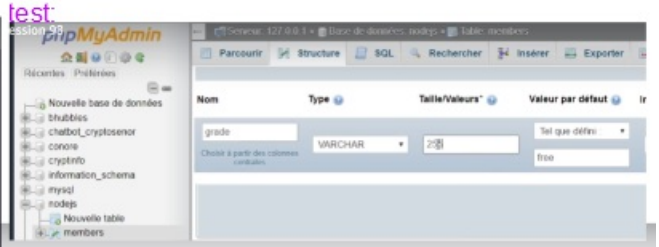
```
<table class="table table-striped">
    <thead>
    <tr>
        {% for index, value in members[0] %}
            <th scope="col">{{ index }}</th>
        {% endfor %}
    </tr>
    </thead>
```

Dans l'entête, on récupère

```
    </thead>
    <tbody>
        {% for member in members %}
            <tr>
                {% for value in member %}
                    <td>{{ value }}</td>
                {% endfor %}
            </tr>
        {% endfor %}
    </tbody>
</table>
```

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

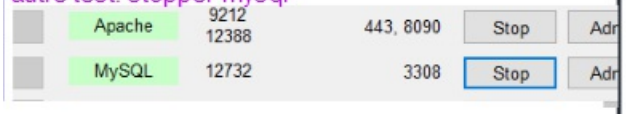
test:



Tous les membres

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

autre test: stopper mysql



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))
})
```

```
// Page d'accueil
app.get('/', (req, res) => {
```

```
// Page récupérant tous les me
app.get('/members', (req, res) => {
```

member.twig
members.twig

```
<title>Membre</title>
```

```
ext-center">Un membre</h1>
```

copie ajustée de members.twig

```
<thead>
<tr>
    {% for index, value in member %}
        <th scope="col">{{ index }}</th>
    {% endfor %}
```

```
<tbody>
    <tr>
        {% for value in member %}
            <td>{{ value }}</td>
        {% endfor %}
```

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()

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

/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!

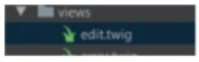
avec max=

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

une petite ternaire!

Page de modifications

Section 13, session 100




```
<title>Modification</ti
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 %}
```

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

Ceci permet d'afficher la page pour éditer

activons le «form»

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

Il faut gérer un post: envoyer 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

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

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

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

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

Testons!

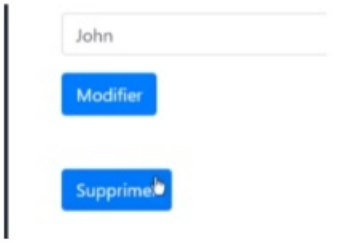
Donc on peut supprimer index.html

101. Méthode de suppression

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

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

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

le input est caché



app.js

```
            res.redirect('/members')
        })
    })

    app.post('/delete', (req, res) => {
        apiCall('/members/'+req.body.id, 'delete', {}, res, () => {
            res.redirect('/members')
        })
    })
```

On redirige vers members

Commentons:

```
// Méthode permettant de supprimer un membre
```

102. Page d'ajout d'un membre

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

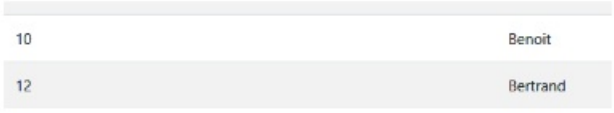
Affichage


```
<title>Créer un membre</tit
ead>

">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>
```

```
//méthode pour créer le membre
app.post('/insert', (req, res) => {
    apiCall('/members/', 'post', {
        name: req.body.name
    }, res, () => {
        res.redirect('/members')
    })
})
```

traitement du formulaire



Liens divers & Conclusion

Section 13, session 103

Pour faire des liens sur members:

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

```
        </tbody>
    </table>
    <a href="/insert" class="btn btn-primary">Ajouter un membre</a>
```

et le bouton d'ajout de 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!