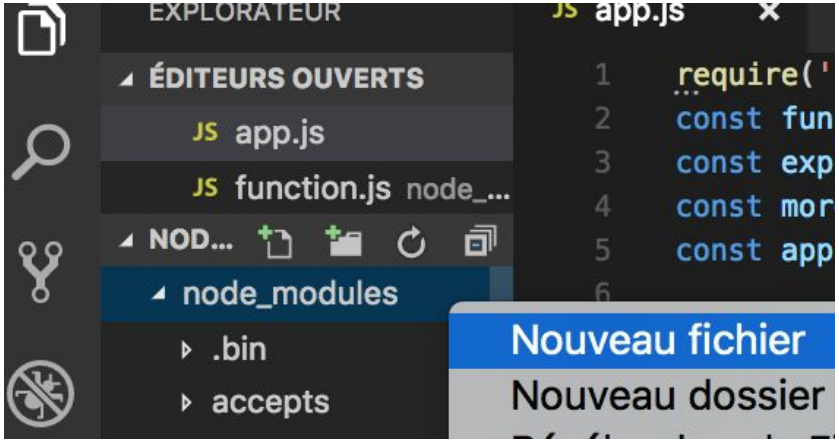
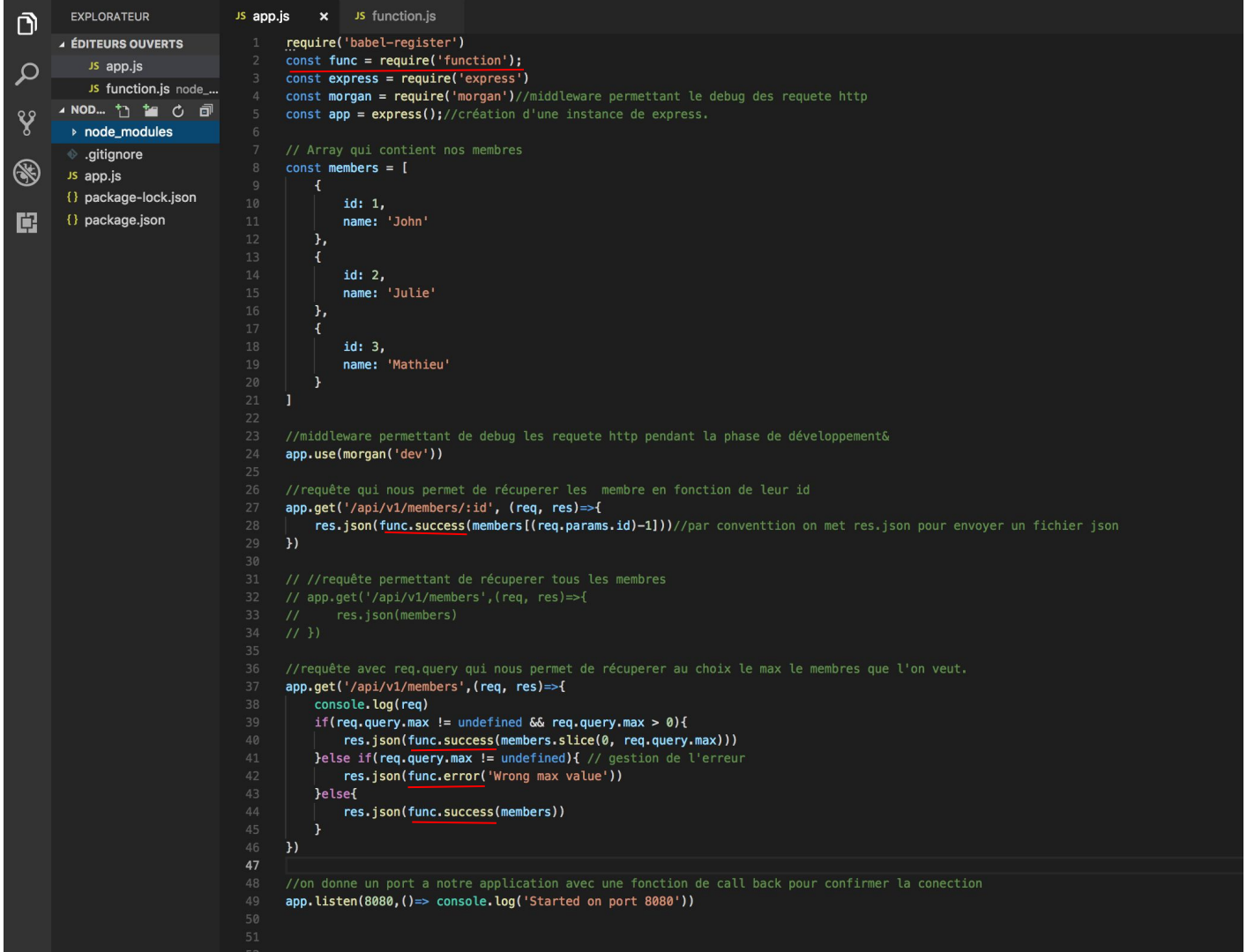
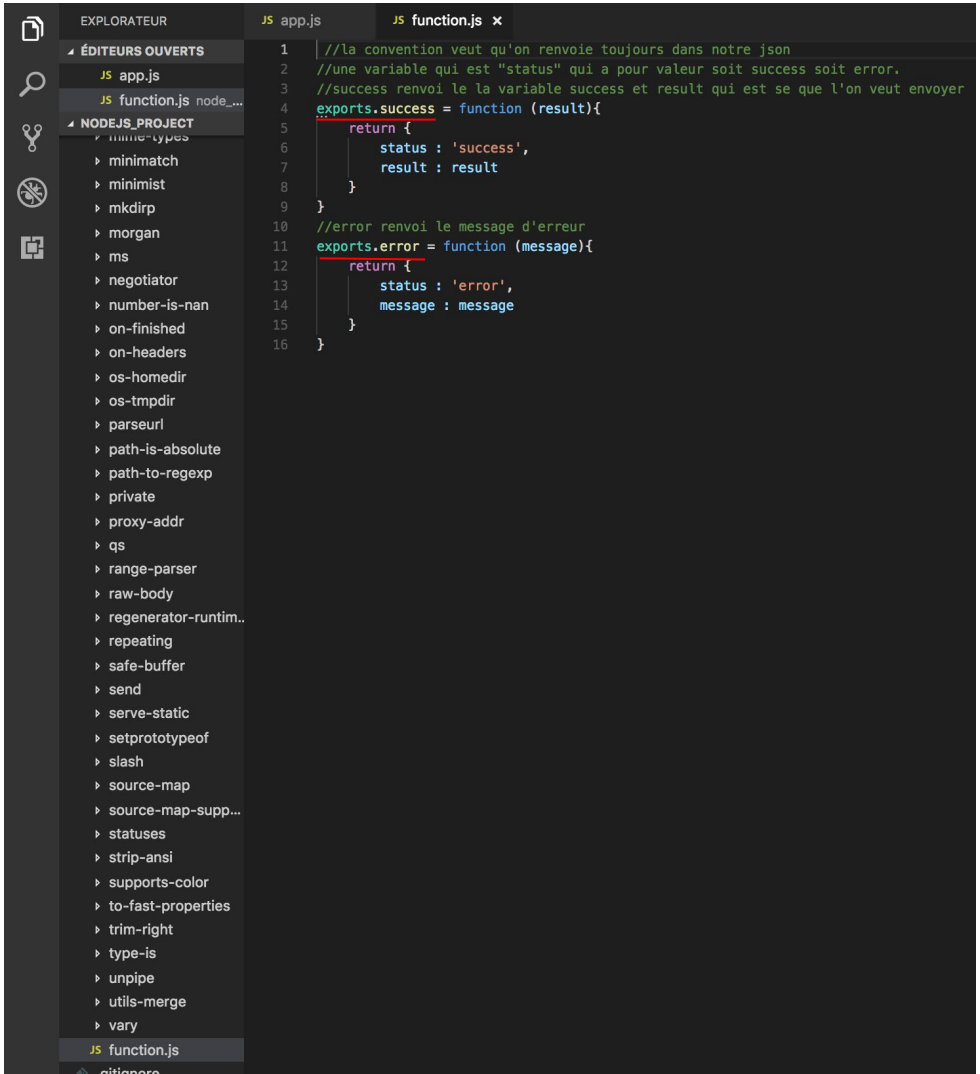
Pour mieux y voir dans notre code on va créer un module qui va contenir toute nos fonction de gestions de success et d'error puis les importer dans app.js et les refactorer pour les utiliser




EXPLORATEUR
ÉDITEURS OUVERTS
app.js
function.js node_...
NOD...
node_modules
.gitignore
app.js
package-lock.json
package.json
app.js
function.js
require('babel-register')
const func = require('function');
const express = require('express')
const morgan = require('morgan')//middleware permettant le debug des requete http
const app = express();//création d'une instance de express.
// Array qui contient nos membres
const members = [
{
id: 1,
name: 'John'
},
{
id: 2,
name: 'Julie'
},
{
id: 3,
name: 'Mathieu'
}
]
//middleware permettant de debug les requete http pendant la phase de développement&
app.use(morgan('dev'))
//requête qui nous permet de récuperer les membre en fonction de leur id
app.get('/api/v1/members/:id', (req, res)=>{
res.json(func.success(members[(req.params.id)-1]))//par conventtion on met res.json pour envoyer un fichier json
})
// //requête permettant de récuperer tous les membres
// app.get('/api/v1/members',(req, res)=>{
// res.json(members)
// })
//requête avec req.query qui nous permet de récuperer au choix le max le membres que l'on veut.
app.get('/api/v1/members',(req, res)=>{
console.log(req)
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){ // gestion de l'erreur
res.json(func.error('Wrong max value'))
}else{
res.json(func.success(members))
}
})
//on donne un port a notre application avec une fonction de call back pour confirmer la conection
app.listen(8080,()=> console.log('Started on port 8080'))


EXPLORATEUR
ÉDITEURS OUVERTS
app.js
function.js node_...
NODEJS_PROJECT
minimatch
minimist
mkdirp
morgan
ms
negotiator
number-is-nan
on-finished
on-headers
os-homedir
os-tmpdir
parseurl
path-is-absolute
path-to-regexp
private
proxy-addr
qs
range-parser
raw-body
regenerator-runtim...
repeating
safe-buffer
send
serve-static
setprototypeof
slash
source-map
source-map-supp...
statuses
strip-ansi
supports-color
to-fast-properties
trim-right
type-is
unpipe
utils-merge
vary
function.js
app.js
function.js
1 //la convention veut qu'on renvoie toujours dans notre json
2 //une variable qui est "status" qui a pour valeur soit success soit error.
3 //success renvoi le la variable success et result qui est se que l'on veut envoyer
4 exports.success = function (result){
5 return {
6 status : 'success',
7 result : result
8 }
9 }
10 //error renvoi le message d'erreur
11 exports.error = function (message){
12 return {
13 status : 'error',
14 message : message
15 }
16 }