Nodejs (V2)

Said Gounane
ISIL 2021-2022

INTRODUCTION

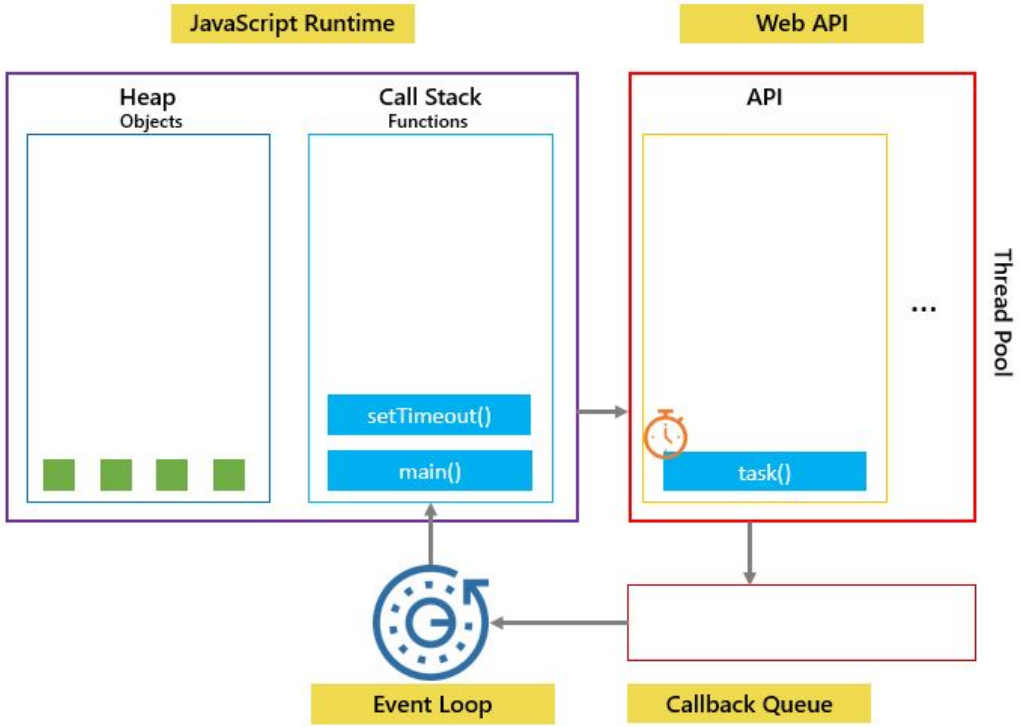
C’est Quoi Node.js

1. Node.je est un environnement côté serveur, open source
2. Node.js est multiplatform
3. Node.js utilise du javascript côté serveur

Pourquoi Node.js

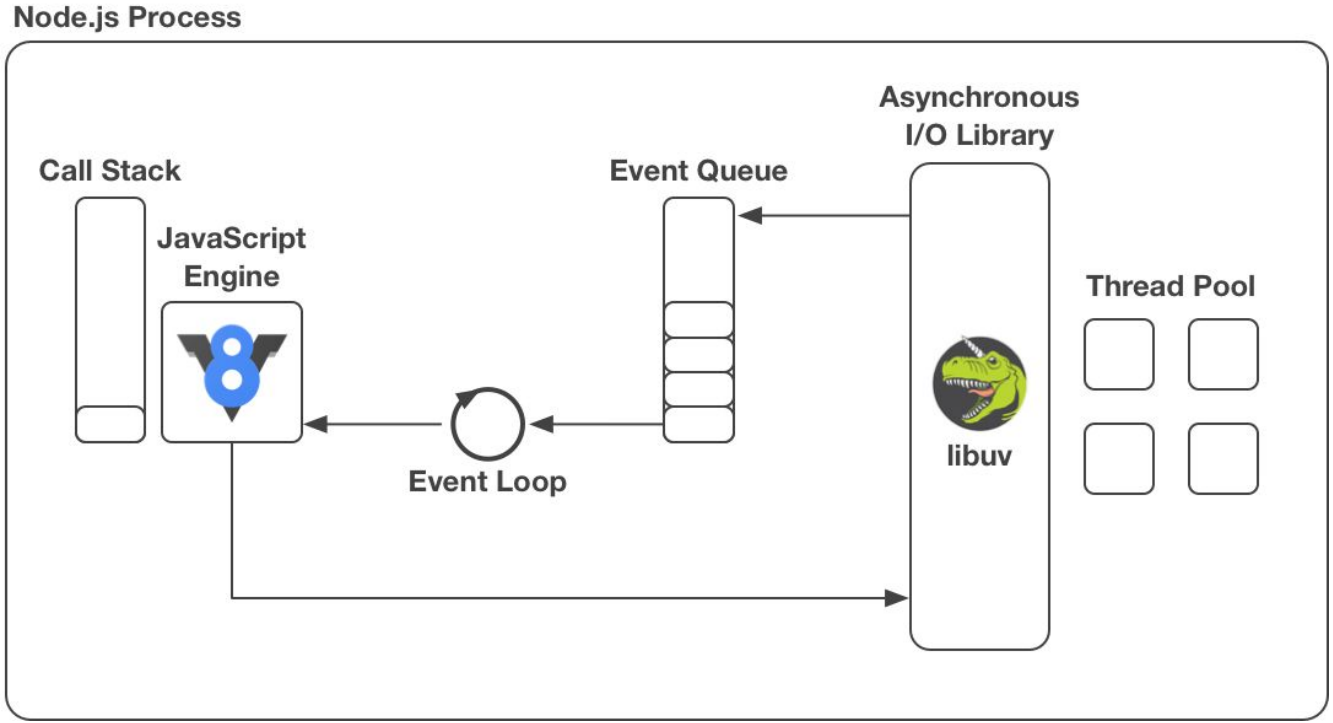
- Node.js utilise une programmation ***Asynchrone***
- Node.js est mono-Thread non-blocking.
- Un programme Node.js exécute les instructions sans arrêt et garantit un fonctionnement non bloquant.
- Node.js utilise le moteur d'exécution ultrarapide V8 de Google Chrome.
- Ce moteur est caractérisé par la compilation JIT (Just In Time). Il transforme le code JavaScript très rapidement en code machine.

Principe (js dans le navigateur)



https://www.javascripttutorial.net/javascript-event-loop/

Principe (node.js coté serveur)



https://itnext.io/javascript-promises-with-node-js-e8ca827e0ea3

Pourquoi Node.js

Pour lire un fichier dans le serveur avec:

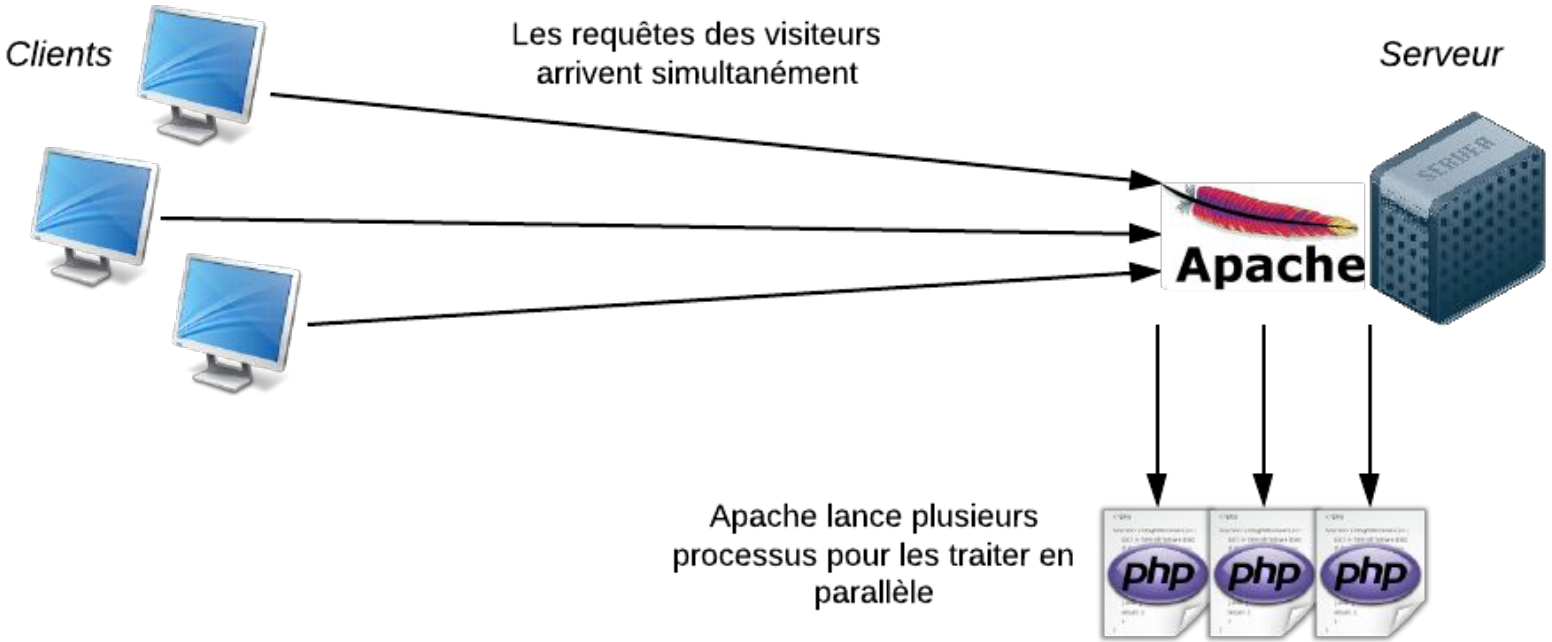
PHP

1. Envoyer la requête au serveur
2. Demander au système de fichier d'ouvrire le fichier
3. **Attendre l'ouverture du fichier**
4. Envoyer le contenue au client
5. Passer à la requête suivante

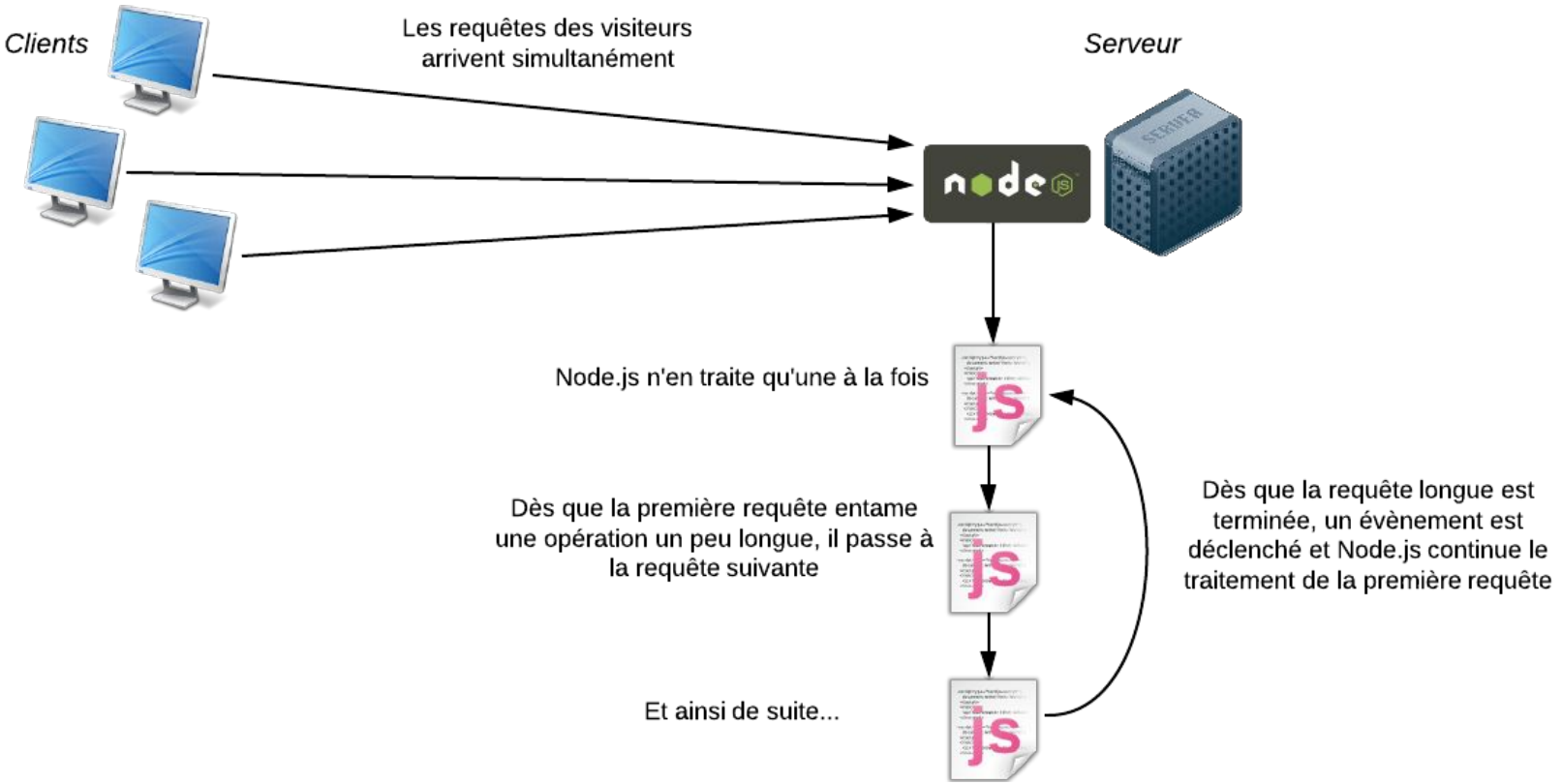
Node.js

1. Envoyer la requête au serveur
2. Demander au système de fichier d'ouvrire le fichier
3. Passer à la requête suivante
4. Quand le contenu du fichier est prêt Envoyer le contenue au client

Pourquoi Node.js



Pourquoi Node.js



Que peut faire Node.js

- Générer des page de contenue dynamique
- Créer, lire, modifier, supprimer des fichiers dans le serveur
- Créer, lire, modifier, supprimer des données dans une base de données
- Accéder au resource matériel du serveur (wifi, bluetooth, webcam ...)
- ...

Nodejs REPL (Read-Eval-Print-Loop)

- Télécharger et installer nodejs 16.18 (https://nodejs.org)
- Dans un terminal tapez ***node***

```
$  node
Welcome to Node.js v18.2.0.
Type ".help" for more information.
> console.log(“Hello nodejs!!!”)
Hello nodejs!!!
undefined
>
```

Premier script Node.js

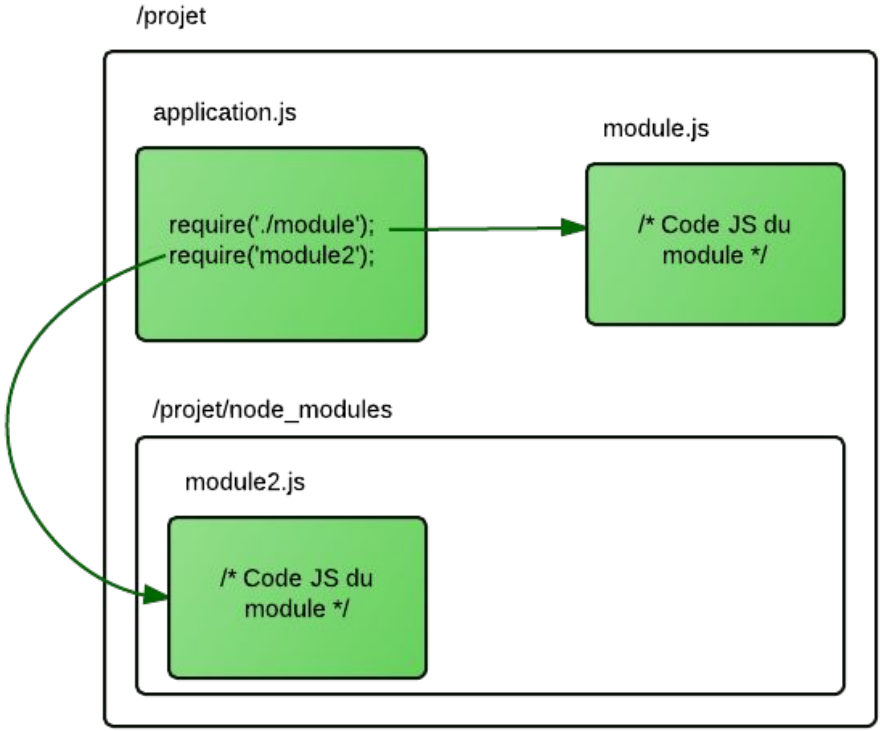
- Dans un fichier hello.js écrire
  - console.log(“Hello Nodejs!!!”)
- Dans un terminal tapez ***node hello.js***

```
$  node hello.js
Hello nodejs!!!
$
```

Les MODULES

Les modules

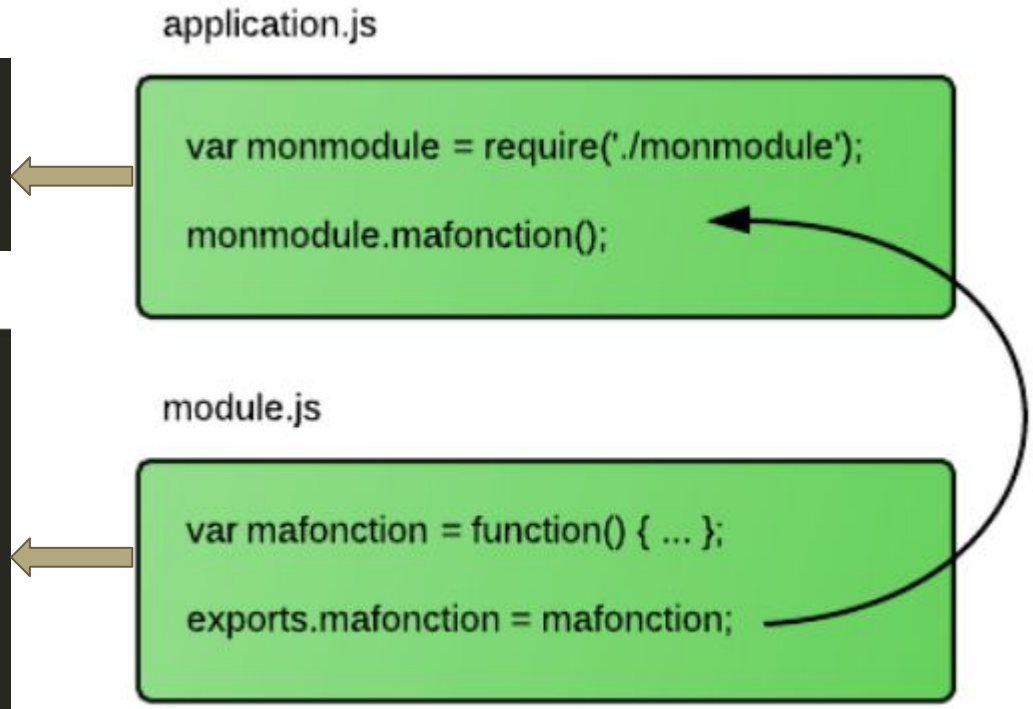
- Le noyau de Node.js est très petit.
- Ne peut pas faire grand chose
- Mais très riche on ***modules***



Les modules

```
1  var monmodule = require('./monmodule');
2
3  monmodule.direBonjour();
4  monmodule.direByeBye();
```

```
1  var direBonjour = function() {
2      console.log('Bonjour !');
3  }
4
5  var direByeBye = function() {
6      console.log('Bye bye !');
7  }
8
9  exports.direBonjour = direBonjour;
10 exports.direByeBye = direByeBye;
```



npm

- Npm : Node Package Manager
- Avec npm on peut découvrir et installer des paquets (packages)
- Un paquet node.js est un répertoire contenant un ou plusieurs modules ou bibliothèques JavaScript.
- Pour initialiser un projet
    - ***$ npm init*** *// dans le dossier du projet*

Les modules

- Chercher un module
  - $***npm search nomModule***
- Installer un module
  - $***npm install nomModule*** //installation locale
  - $***npm install nomModule -g*** //installation globale
- Mettre à jour un module
  - $***npm update nomModule***
- Publier un module
  - $npm adduser //Creer un compte su npm
  - npm publish // publier le module

Les modules dans nodejs

Dans node.js on trouve trois types de modules:

- Module intégrés dans le corps du node.js
  - Inclusion directe
  - `const http = require('http');`
- Module tiers
  - Installation du module : $npm install express
  - Puis inclusion : `const express = require('express');`
- Module pérsonele
  - Création du module
  - Inclusion du module avec précision du chemin vers ce module
  - `const http = require('./monDossier/monModule');  //Omission de l'extension .js`

Créer un module node.js

- Tout fichier node.js peut être considéré comme un module si on exporte ses fonctions et/ou ses données.
- Écrire des modules vous permettra de contribuer à la communauté Node.js.
- Tous les packages que vous utilisez sur npm ont été regroupés et partagés sous forme de modules.
- La création de modules est une compétence essentielle pour un Développeur Node.js

Créer un module node.js

- Pour créer un module, il faut:
  - $ *mkdir monModule* //Créer un dossier pour le module
  - $ *cd monModule*
  - $ *npm init -y* //initialiser package.json
  - $ *touch index.js* //créer le fichier d'entré du package
  - $ *nano index.js* //éditer le fichier et exporter des fonctions et/ou des données

package.json

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

Créer un module node.js

monModule/index.j

s

```
class RandomAge{
    constructor(minAge,maxAge){
        this.minAge=minAge;
        this.maxAge=maxAge
    }
    getRandomAge(){
        return Math.floor(Math.random()*(this.maxAge-this.minAge+1)+this.minAge)
    }
}
module.exports=RandomAge
//exports.RandomAge=RandomAge
```

Créer un module node.js

monModule/test.js

```
const RandomAge=require("./index")
// dans le cas de exports.RandomAge=RandomAge dans index.js
Const {RandomAge}=require("./index")
const ra=new RandomAge(20,60)
console.log(`MinAge: ${ra.minAge}`)
console.log(`MaxAge: ${ra.maxAge}`)
console.log(`Random Age: ${ra.getRandomAge()}`)
```

Installer le module

- Dans un nouveau projet monProjet:
  - $mkdir monProjet
  - $cd monProjet
  - $npm init -y
  - $npm install ../monModule
  - $touch index.js
- Importer le module ***monModule***

monProjet/index.js

```
const RandomAge=require("monModule")
const ra=new RandomAge(20,60)
console.log(`MinAge: ${ra.minAge}`)
console.log(`MaxAge: ${ra.maxAge}`)
console.log(`Random Age: ${ra.getRandomAge()}`)
```

package.json

```
{
 "name": "monprojet",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "scripts": {
   "test": "echo \"Error: no test specified\" && exit 1"
 },
 "keywords": [],
 "author": "",
 "license": "ISC",
 "dependencies": {
   "monmodule": "file:../monModule"
 }
}
```

Node.js et le système de fichiers

Introduction

- Travailler avec des fichiers et des répertoires est l'un des besoins fondamentaux d'une application full-stack.
  - téléverser des images, des CV ou d'autres fichiers vers un serveur.
  - lire des fichiers de configuration,
  - Créer, lire, supprimer, déplacer des fichiers, ou même changer leurs permissions
- Le module de système de fichiers de Node.js fournit plusieurs API pour interagir avec les systèmes de fichiers de manière transparente.
- La plupart des API sont personnalisables avec des options et des drapeaux.
- On peut effectuer des opérations de fichiers synchrones et asynchrones.

Création et écriture

Pour écrire dans un fichier on peut utiliser les méthodes suivantes:

les versions ***Asynchrones***

- open(“pathToFile”,”r/w/a”,calleBack)
- append(“pathToFile”,”encoding”,data,calleBack)
- writeFile(“pathToFile”,”encoding”,data,calleBack)
- unlink(“pathToFile”,calleBack)
- rename(“pathToFile”,”newName”,calleBack)

Ou les versions ***synchrones***

- openSync(“pathToFile”,”encoding”,”r/w/a”,calleBack)
- appendSync(“pathToFile”,”encoding”,data,calleBack)
- writeFile(“pathToFile”,”encoding”,data,calleBack)
- ...

Premier script Node.js

fileReader.js

```
1  var fs = require('fs'),
2    path = './fichier.txt';
3
4  console.log('before');
5
6  fs.readFile(path, function(err, file) {
7    console.log('during');
8    console.log('' + file);
9  });
10
11 console.log('after');
```

- ***Resultat !!!***

Premier script Node.js

fileReader.js

```
1  var fs = require('fs'),
2    path = './fichier.txt';
3
4  console.log('before');
5
6  fs.readFile(path, function(err, file) {
7    console.log('during');
8    console.log('' + file);
9  });
10
11 console.log('after');
```

- ***Resultat !!!***

```
gnu@gnu-ThinkPad-T420s:~/EST/ISIL/WEB/Nodejs/01$ node fileReader.js
before
after
during
ceci est un texte de './fichier.txt'
```

Serveur web avec le module http du Node.js

C’est quoi HTTP

- **H**yper **T**ext **T**ransefer **P**rotocol
- Assure la Communication entre un client et un serveur
- Essentiellement sous forme de Requête/Réponse
- Chaque requête est indépendante des précédentes (Stateless)
- Charger des pages, envoyer des formulaires ...

C’est quoi HTTPS

- **H**yper **T**ext **T**ransefer **P**rotocol **S**ecure
- Echange des données cryptées
- SSL (Secure Sockets Layer) /TLS (Transfer Layer Scurity)
- Il faut installer une certificat sur le serveur

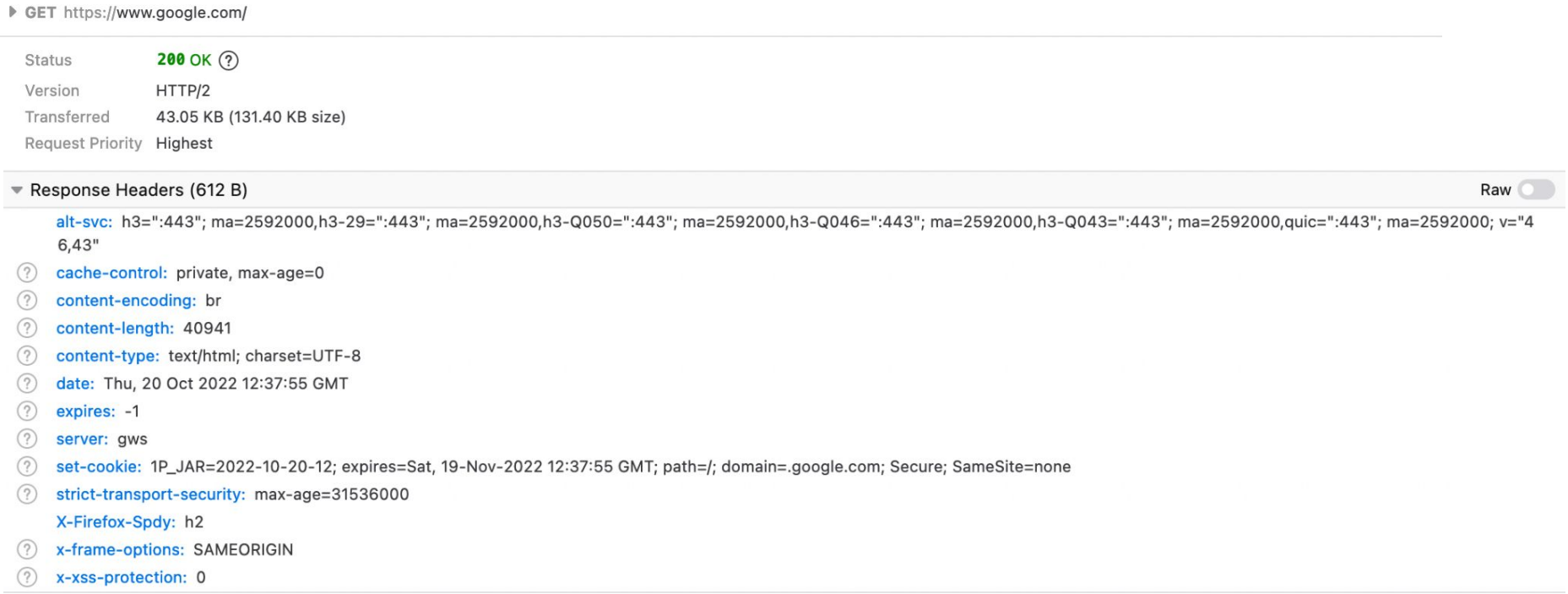
HTTP: Méthodes

- GET: récupérer des données du serveur
- POST: envoyer des données au serveur
- PUT: mettre à jour des données sur le serveur
- DELETE: supprimer des données du serveur

HTTP : Header


Request Headers (804 B)
Raw
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Cookie: 1P_JAR=2022-10-20-12; AEC=AakniGMyj6zoHKAjDRmDfX5Hd3gctDWSL_1f43WA1t2m6n77swJzmmLFLA; NID=511=E7loqCuG-9dh6cq8dl8sLJ6JEIL3YCDBHzIvif_YDnkW84irDX9WZ6NF
gwiiGWsqaMgzQR-wW2rEOlCib3e13RaC3fN9HCmnE8Ru9j4c6A-YIZ9wl-Yxmbiu9M56hmTzdAvwrz2Y7a9aXfW18QXx6ds0iHgYK3xmcgtFjIE9on0; ANID=AHWqTUmGdJzSLn8w8l5Ib929NGrdJf1ov
JdpK7tnf9hDLUiiQJtkZ7B9FVT3vrjt
DNT: 1
Host: www.google.com
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
TE: trailers
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:105.0) Gecko/20100101 Firefox/105.0

HTTP : Header


GET https://www.google.com/
Status 200 OK
Version HTTP/2
Transferred 43.05 KB (131.40 KB size)
Request Priority Highest
Response Headers (612 B)
Raw
alt-svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
cache-control: private, max-age=0
content-encoding: br
content-length: 40941
content-type: text/html; charset=UTF-8
date: Thu, 20 Oct 2022 12:37:55 GMT
expires: -1
server: gws
set-cookie: 1P_JAR=2022-10-20-12; expires=Sat, 19-Nov-2022 12:37:55 GMT; path=/; domain=.google.com; Secure; SameSite=none
strict-transport-security: max-age=31536000
X-Firefox-Spdy: h2
x-frame-options: SAMEORIGIN
x-xss-protection: 0

HTTP: Status Code

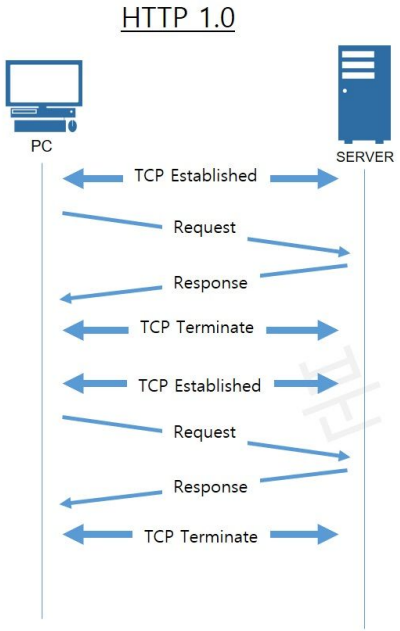
- 100 - 199 : Les réponses informatives,
- 200 - 299 : Les réponses de succès ,
- 300 - 399 : Les messages de redirection,
- 400 - 499 : Les erreurs du client,
- 500 - 599 : Les erreurs du serveur.

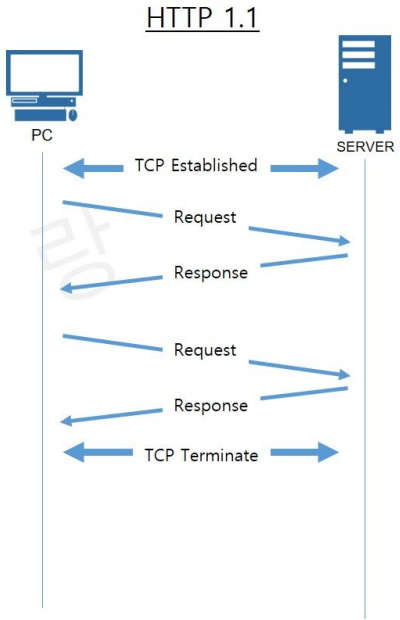
https://developer.mozilla.org/fr/docs/Web/HTTP/Status

HTTP: Status Code

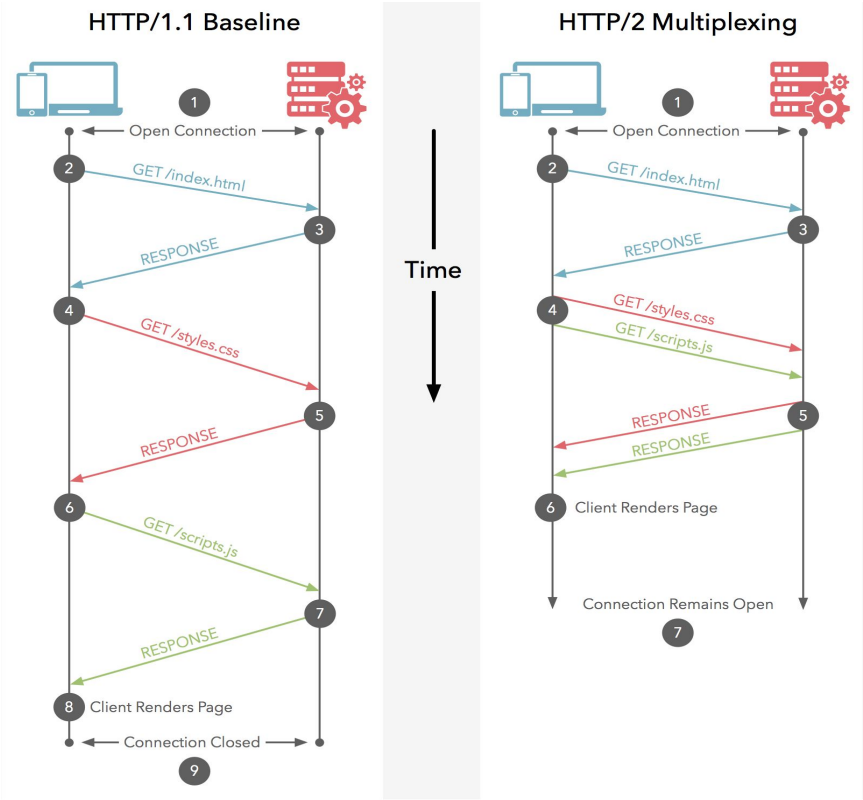
- 200 : OK
- 201 : OK Created
- 301 : Moved Permanently
- 304 : Not Modified
- 400 : Bad Request,
- 401 : Unauthorized
- 403 : Forbidden
- 404 : Not Found
- 405 : Method Not Allowed
- 500 : Internal Server Error
- 502 : Bad Gateway

HTTP 1.0 Vs HTTP 1.1





HTTP 1.1 Vs HTTP 2



Premier serveur web avec Node.js

app.js

```
const http=require("http");
const host="localhost";
const port=3000;
function requestHandler(req,res) {
   res.writeHead(200);
   res.write("<h1>HI!!</h1>");
   res.end();
   //res.end("<h1>H1</h1>")
}
const server=http.createServer(requestHandler)
server.listen(port,host,()=>console.log(`Server is running on http:// ${host}:${port} `))
```

- Pour lancer l'application (le serveur HTTP)
  - $ ***node app.js***
- Pour visualiser le résultat dans le navigateur
  - ***http://localhost:3000/***

Envoyer différents types de contenu

- La réponse renvoyée d'un serveur Web peut prendre divers formats.
  - JSON; HTML ; XML ; CSV
  - PDF ; des fichiers compressés ; images ; audio ; vidéo.
- Pour envoyer un contenu dans node.js, il faut faire deux choses :
  - Définir l'en-tête ***Content-type*** dans les réponses HTTP avec le valeur appropriée.
  - S’assurer que ***res.send()*** obtient les données dans le bon format.

Envoyer différents types de contenu

- ***Content-type*** peut être:
  - Du texte brut : **text/plain**
  - Du HTML : **text/html**
  - Du CSS : **text/css**
  - Une image JPEG : **image/jpeg**
  - Une vidéo MPEG4 : **video/mp4**
  - Un fichier ZIP : **application/zip**
  - etc.

Envoyer du JSON

```
const http=require("http");
const host="localhost";
const port=3000;
function requestHandler(req,res) {
    res.setHeader("Content-type", "application/json")
    res.writeHead(200);
    res.end(JSON.stringify({nom:"gounane",age:22}))
}
const server=http.createServer(requestHandler)
server.listen(port,host,()=>console.log(`Server is running on http://${host}:${port} `))
```

Envoyer du HTML

```
const http=require("http");
const host="localhost";
const port=3000;
function requestHandler(req,res) {
   res.setHeader("Content-type", "text/html")
   res.writeHead(200);
   res.end("<h1> Bonjour Tous!!<h1>")
}
const server=http.createServer(requestHandler)
server.listen(port,host,()=>console.log(`Server is running on http://${host}:${port} `))
```

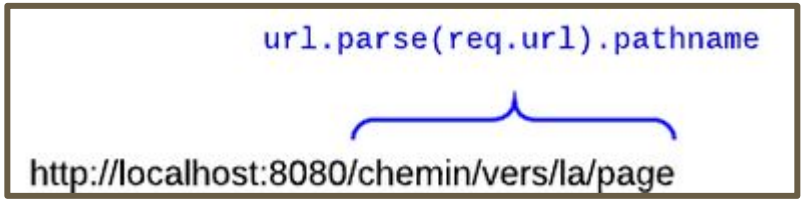
Envoyer une page HTML depuis un fichier

```
const http=require("http");
const fs=require("fs")
const host="localhost";
const port=3000;
function requestHandler(req,res) {
   fs.readFile(__dirname + "/index.html",(err,data)=>{
       res.setHeader("Content-type", "text/html")
       res.writeHead(200);
       console.log(data)
       res.end(data)
   })
}
const server=http.createServer(requestHandler)
server.listen(port,host,()=>console.log(`Server is running on http:// ${host}:${port} `)
```

Node.js: Gestion des requêtes

Pour récupérer la page demandée par le visiteur:

1. on utilise le module "***url***" pour décomposer le url de la requête
2. Extraire Le chemin de la ressource en question (***pathname***)
3. Envoyer la réponse adequate



Node.js: Gestion des requêtes

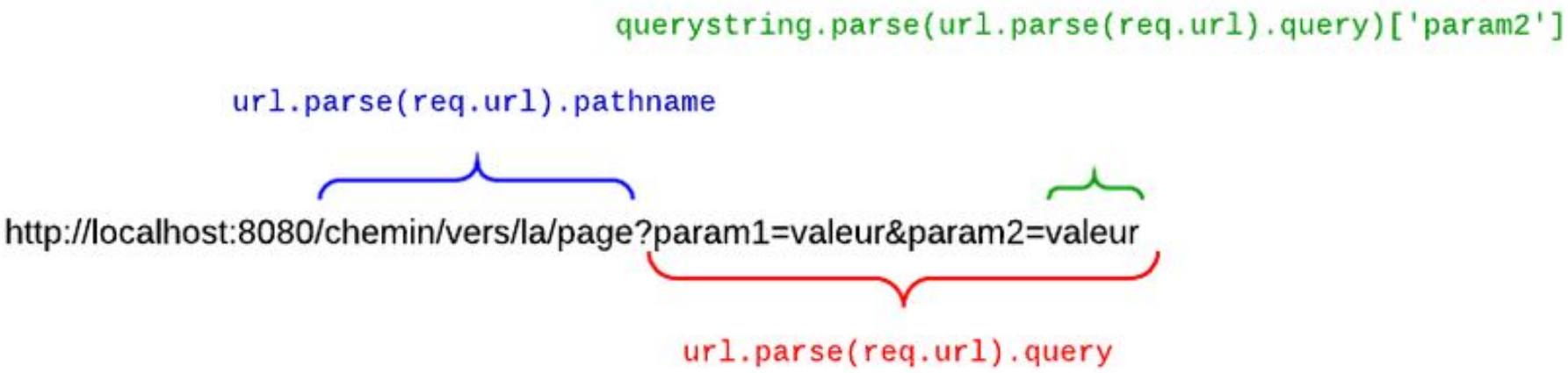
```
1  var http = require('http');
2  var url = require('url');
3
4  var server = http.createServer(function(req, res) {
5      var page = url.parse(req.url).pathname;
6      console.log(page);
7      res.writeHead(200, {"Content-Type": "text/plain"});
8      if (page == '/') {
9          res.write('Accueil');
10     }
11     else if (page == '/contact') {
12         res.write('Nous contscter');
13     }
14     else if (page == '/cours/web/nodejs') {
15         res.write('Node.js is greate');
16     }
17     res.end();
18 });
19
20 server.listen(3000);
```

Node.js: Gestion des requête

Pour récupérer les paramètres de la requête:

1. Utiliser le module "***url***" pour décomposer le url de la requête
2. Extraire Les paramètres en question (***query***) => chaîne de caractères
3. Utiliser le module "***querystring***" pour décomposer cette chaîne en {key: value}
4. Envoyer la réponse adequate

Node.js: Gestion des requête



Node.js: Gestion des requête

```
var http = require('http');
var url = require('url');

var querystring = require('querystring');

var server = http.createServer(function(req, res) {
    var params = querystring.parse(url.parse(req.url).query);
    res.writeHead(200, {"Content-Type": "text/plain"});
    if ('prenom' in params && 'nom' in params) {
        res.write('Vous vous appelez ' + params['prenom'] + ' ' + params['nom']);
    }
    else {
        res.write('Il manque des parametres!!');
    }
    res.end();

});

server.listen(3000);
```

1. Lancer le serveur: ***$ node params.js***
2. Envoyer la requête : ***http://localhost:3000/?nom=yahyaoui&prenom=anas***
3. Puis la requête: ***http://localhost:3000***

Node.js: les evenements

- On Node.js, un grand nombre d'objets émettent des évènements.
- Ces objets héritent tous d'un objet EventEmitter fourni par Node.
- Par exemple L'objet **server** de **HTTP** possède les événements:
  - ***Request***
  - ***Connection***
  - ***Close***
  - ***Connect***
  - ***Upgrade***
  - ***clienError***

Node.js: les evenements

- Pour écouter à un événement sur un objet on appel la méthode on() de cet objet:

*obj.**on**(‘**evenement**’, **callbackFunction**)*

Node.js: les evenements

- Pour écouter à un événement sur un objet on appel la méthode on() de cet objet:

*obj.**on**(‘**evenement**’, **callbackFunction**)*

```
var server = http.createServer();
server.on('request', function(req, res) { });
```

```
var server = http.createServer(function(req, res) { });
```

Node.js: les evenements

```
1  var http = require('http');
2
3  var server = http.createServer(function(req, res) {
4    res.writeHead(200);
5    res.end('Salut ISIL !');
6  });
7
8  server.on('close', function() { // On écoute l'évènement close
9      console.log('Au revoir!!');
10 });
11
12 server.listen(3000); // Démarre le serveur
13 server.close(); // Arrête le serveur. Déclenche l'évènement close
14
```

Node.js: les evenements

- Pour créer un objet susceptible d'émettre un événement:
  - Inclure le module '***events***'
  - Créez un objet basé sur EventEmitter
  - Quelque part dans l'app, Pour écouter l'événement sur cet objet
    - *obj.**on**('**nomEvent**', **function(param1,param2, ...){ ... }**)*
  - Pour déclencher l'événement on appele la méthode ***emit('nomEvent', param1, param2, ...)***

Node.js: les evenements

```
1  var EventEmitter = require('events').EventEmitter;
2
3  var monObjet = new EventEmitter();
4
5  monObj.on('monEvent', function(param){// on ecoute monEvent
6      console.log(param);
7
8  });
9
10 monObje.emit('monEvent', 'Un texte de monEvent');// Declancher monEvent
```