WIM - M4103C

Nodejs

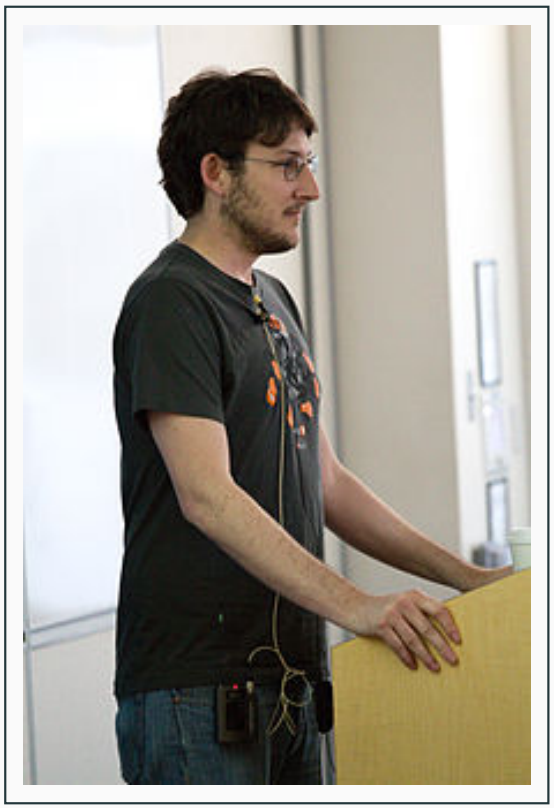
monnerat@u-pec.fr

IUT de Fontainebleau

Sommaire

Introduction

Nodejs ?



Attention

Ce n'est pas un framework !

Programme (environnement) créé par Rayn Dahl en 2009, basé sur le moteur V8 javascript de Google.

- Utilise javascript comme langage.
- Orientée vers les applications réseaux.
- Programmation événementielle.
- Modéle asynchrone.
- Boucle d'événement unique pour tous les clients.

Javascript

Évenementiel

Hors du navigateur

Asynchrone

Avec des fonctions pour serveur

Utilise V8

Exemple : un (petit) serveur Web

Le code Javascript

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(8080, '127.0.0.1');
console.log('Server running at http://127.0.0.1:8080/');
```

Exécution du programme

```
>node example.js
Server running at http://127.0.0.1:8080/
```

Programmation événementielle

Evénements

```
objet.on('evt',function(args){
/* je traite l'événement */
});
```

```
objet.emit('evt',args);
/* emission d'un
 * événement
 */
```

Modèle Asynchrone

Approche synchrone

```
let rep1 = RequeteLongue();
let rep2 = AutreRequeteLongue();
```

La deuxième fonction commence après la terminaison de la première

Cela peut être très pénalisant notamment si la première fonction est en attente d'E/S.

Approche asynchrone

```
RequeteLongue(function(err,data){
  if (err){
    /* mince !*/
  }else{
    /* youpi : data ....*/
  }
});
AutreRequeteLongue(function(err,data){
  if (err){
    /* mince !*/
  }else{
    /* youpi : data ...*/
  }
});
```

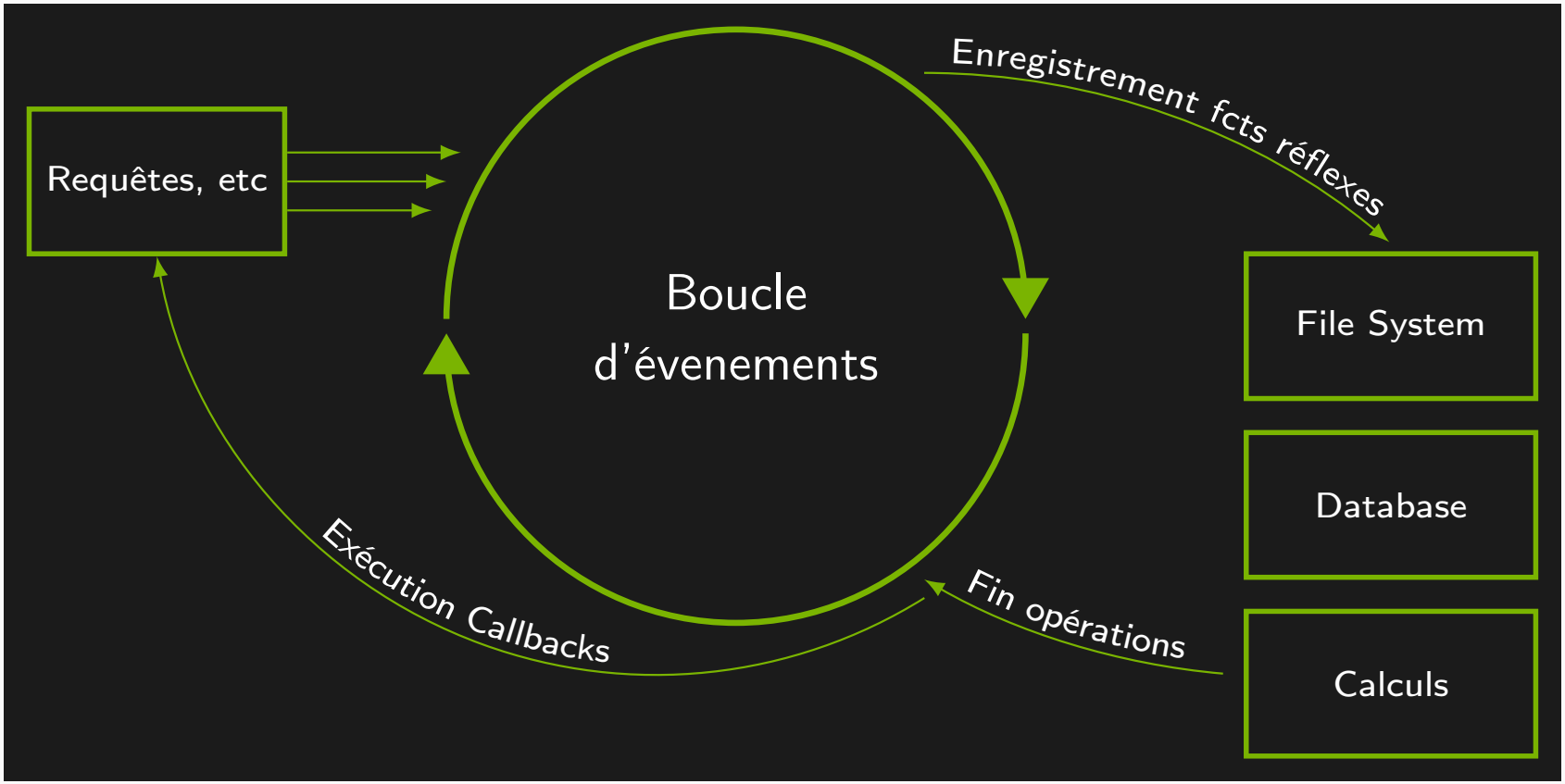
Une fonction réflexe (Callback) est exécutée quand le résultat est disponible.

Penser Callback

```
UneFonction(args,function(err,data){
  if (err){
    /* gestion cas d'erreurs */
  }else{
    /* data contient
     * le résultat
     */
  }
});
```

La plupart des librairies de nodejs utilise cette technique

Implantation



Un seul thread exécute la boucle.

Ce modèle est utilisé par d'autres programmes :

- Le serveur http `Nginx`.
- La couche graphique X11 d'Unix.

Comparaison de performances

On compare les performances, en situation à peu près identique de :

- Un serveur Apache.
- Un server nodejs

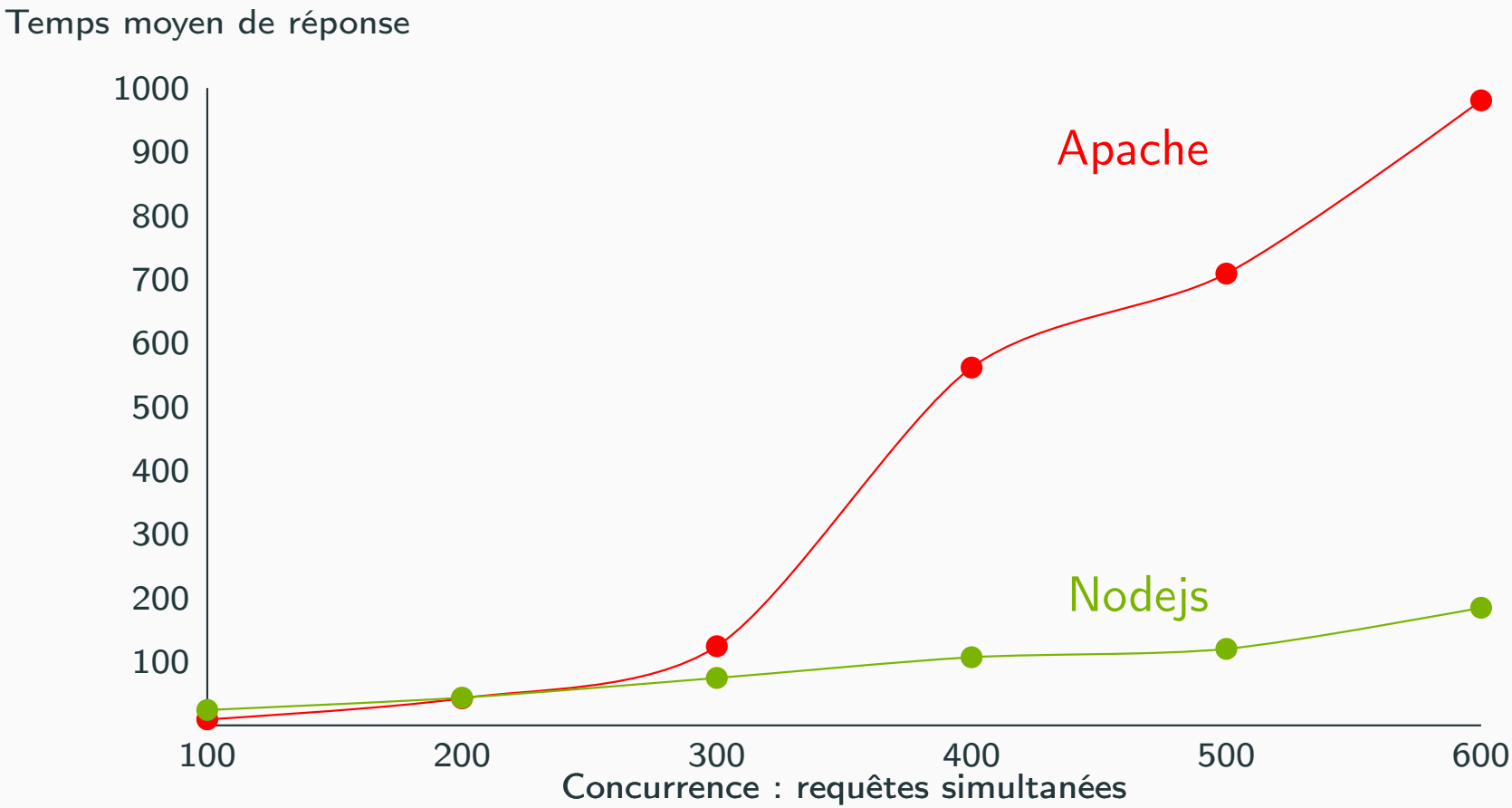
```
ab -n 1000 -c 200 http://127.0.0.1/hello.html
```

permet de mesurer le nombre de requêtes satisfaites par seconde.

Le code du serveur nodejs

```
var http = require('http');
var fs= require('fs');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  var s=fs.createReadStream('./index.html');

  s.on('open',function(){
    s.pipe(res);
  })
  s.on('end',function(){
    res.end();
  });
}).listen(8080, '127.0.0.1');
console.log('Server running at http://127.0.0.1:8080/');
```


Temps moyen de réponse
1000
900
800
700
600
500
400
300
200
100
Apache
Nodejs
100
200
300
400
500
600
Concurrence : requêtes simultanées

Ecosystème

Beaucoup de librairies intégrées

Timers	Process	Events	Buffers
Streams	Crypto	File System	REPL
Net	HTTP	HTTPS	TLS/SSL
DNS	UDP/Datagram	URL	Path

```
var http = require('http');
var fs = require('fs');
var dns = require('dns');
```

Npm

`npm` est un gestionnaire de paquets pour nodejs. `https://www.npmjs.com/`

Beaucoup de paquets disponibles :

- Express : framework web minimaliste.
- Socket.io : framework Serveur "temps réel" (WebSocket).
- etc.

Ecosystème

Express

Express

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World')
});

app.listen(3000);
```

Routage basique

```
// respond with "Hello World!" on the homepage
app.get('/', function (req, res) {
  res.send('Hello World!');
})
// accept POST request on the homepage
app.post('/', function (req, res) {
  res.send('Got a POST request');
})
// accept PUT request at /user
app.put('/user', function (req, res) {
  res.send('Got a PUT request at /user');
})
// accept DELETE request at /user
app.delete('/user', function (req, res) {
  res.send('Got a DELETE request at /user');
})
```

Template

1. Express a besoin de savoir ou sont les templates des vues :

```
app.set('views', __dirname+'/views');
```

2. Express a besoin de connaître le moteur de template à utiliser :

```
app.set('view engine', 'jade');
```

3. On crée un template jade

```
html
  head
    title!= title
  body
    h1!= message
```

4. On crée la route avec la vue précédente comme réponse.

```
app.get('/', function (req, res) {
res.render('index', { title: 'Hey', message: 'Hello there!'});
})
```

WebSocket et Nodejs

WebSocket est une technologie "évoluée" qui permet d'ouvrir un canal de communication interactif entre un navigateur (côté client) et un serveur. Avec cette API vous pouvez envoyer des messages à un serveur et recevoir ses réponses de manière événementielle sans avoir à aller consulter le serveur pour obtenir une réponse (polling).

Le protocole WebSocket est décrit dans la RFC 6455 :

`http://tools.ietf.org/html/rfc6455`

```
[Constructor(DOMString url, optional (DOMString or DOMString[]) protocols)]
interface WebSocket : EventTarget {
  readonly attribute DOMString url;
  // ready state
  const unsigned short CONNECTING = 0;
  const unsigned short OPEN = 1;
  const unsigned short CLOSING = 2;
  const unsigned short CLOSED = 3;
  readonly attribute unsigned short readyState;
  readonly attribute unsigned long bufferedAmount;
  // networking
           attribute EventHandler onopen;
           attribute EventHandler onerror;
           attribute EventHandler onclose;
  readonly attribute DOMString extensions;
  readonly attribute DOMString protocol;
  void close([Clamp] optional unsigned short code, optional DOMString reason);
  // messaging
           attribute EventHandler onmessage;
           attribute DOMString binaryType;
  void send(DOMString data);
  void send(Blob data);
  void send(ArrayBuffer data);
  void send(ArrayBufferView data);
};
```

Api coté client très simple.

```
var wsUri = "ws://echo.websocket.org/";
websocket = new WebSocket(wsUri);
websocket.onopen = function(evt) { onOpen(evt) };
websocket.onmessage = function(evt) { onMessage(evt) };
websocket.onclose = function(evt) { onClose(evt) };
websocket.onerror = function(evt) { onError(evt) };
message = "on est le " +  new Date().toString();
websocket.send(message);
```

Il existe plusieurs librairies implantant cette technologie pour nodejs :

- Socket.IO : `http://socket.io/`
- WebSocket-Node : `https://github.com/theturtle32/WebSocket-Node`

Socket.IO

Le client `hello.html`

```
var socket;
socket = io.connect('http://localhost:3000');

socket.on('ok',function(data){
  $("#reponse").empty().text(data.message);

  socket.emit('message',{
    "message":'Salut aussi !'
  });

});
```

Le serveur

```
var app = require('express')();
var http = require('http').Server(app);
var io = require('socket.io')(http);

app.get('/', function(req, res){
  res.sendfile("./hello.html");
});

io.on('connection', function(socket){
  console.log('a user connected');
  socket.on('message',function(data){
    console.log("message "+data.message);
  });
  socket.emit('ok',{message:'Salut'});
});

http.listen(3000, function(){
  console.log('listening on *:3000');
});
```