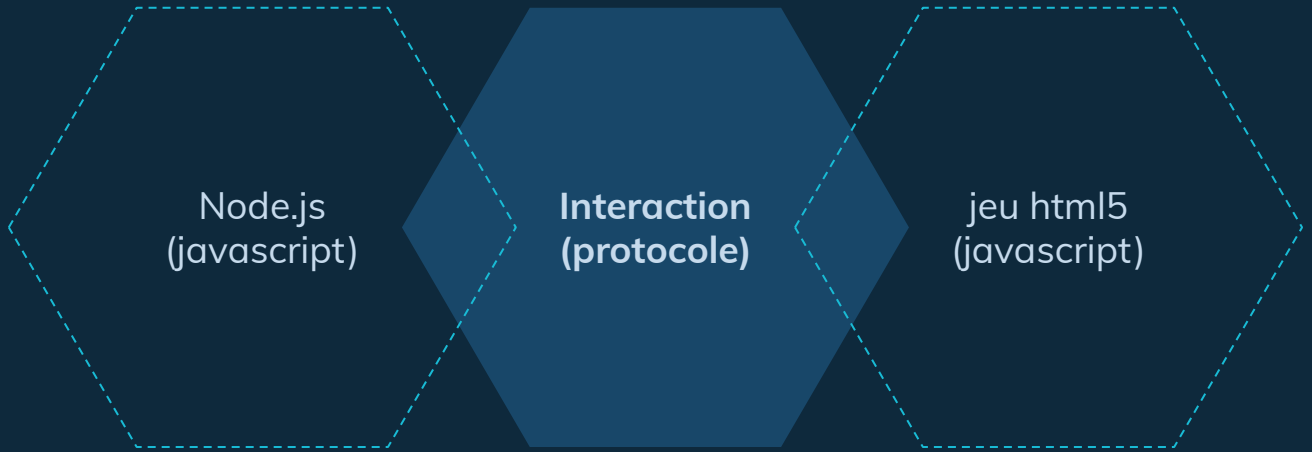


Node.js

Notes de cours
par Nadine
Giasson
St-Amand



Idée générale de node.js





Utilisations de node.js

Survol des chiffres et des applications

3,000,000

d'utilisateurs

2,000,000 tél

Téléchargements par mois

2,000,000,000

Installation paquetage par mois

98%

Des compagnies Fortune 500 l'utilisent régulièrement

NODE IS IN PRODUCTION AT BIG COMPANIES

Big companies have moved from experimenting with Node to deploying it into production. Companies like...































Companies Using Node



"We use Node.js to power the back-end of our real-time collaboration software. Between its superb scalability and performance, Node is the only logical choice for our current and future applications."

Pete Fredricks
Co-Founder, CEO









http://nodejs.org/industry/

meetup

Meetups (foyers)



nodeJS Meetup Groups



Groups	Members	Interested	Cities	Countries
482	128,750	12,919	278	62



Installation



Étapes présentées dans ce guide





1 Téléchargement

https://nodejs.org/en/download/



2

Installation

Pour Window :
https://drive.google.com/open?id=0Bwi_qZJNZ9UqWVNPOGdRbVpPU0U

Pour Linux

Placer dans un répertoire à la racine appelé 'nodejs'



2

https://nodejs.org/en/download/package-manager/#debian-and-ubuntu-based-linux-distributions
https://github.com/nodesource/distributions/blob/master/README.md

```
curl -fsSL https://deb.nodesource.com/setup_19.x | sudo -E bash - &&\
sudo apt-get install -y nodejs
```

Installation

Commandes dans Linux :

https://nodejs.org/en/ (lts)

```
mv Downloads/node-v8.12.0-linux-x64.tar.xz .
```

```
tar xJf node-v8.12.0-linux-x64.tar.xz
```

2

Installation

Commandes dans Linux :

sudo apt-get install nodejs

sudo apt-get install npm

npm est le paquet manager (comme apt, comme nuget, comme chocolatay)



3 Premiers essais

Les hello worlds ...



Hello World





Aspects techniques

Toutes sortes d'hello world



Allo !

Voici le 'hello world'

-

Ligne de commande

Il sera composé de fichiers serveurs exécutés en ligne de commande comme en console



Hello World





Programme 'hello world' en mode interactif (shell)

Entrée dans le shell

cd .. (répété)

C:\>cd nodejs
C:\nodejs>node
>

Hello World

> console.log('Allo !');
Allo !





Programme 'hello world' en ligne de commande

Fichier coucou.js

```
var nom = 'nadine';
console.log('coucou ' + nom);
```

Invocation

cd .. (répété)

C:\>cd nodejs

C:\nodejs>node coucou

Programme 'hello world' en tant que serveur

Fichier serveur-web.js

```
var http = require('http');
var repondeur = function(requete,reponse)
{
        console.log( 'execute response' );
}
var serveur = http.createServer(repondeur);
serveur.listen(3000);
```

- Lancer le service

C:\>cd nodejs

C:\nodejs>

nodejs **serveur-web**

–

- Autoriser le firewall
- Tester avec le fureteur à http://localhost:3000/

https://nodejs.org/api/https.html et https://nodejs.org/api/http.html

Programme 'hello world' en tant que serveur

Fichier allo-clients.js

```
var http = require('http');
var repondeur = function(requete,reponse){

        reponse.writeHead(200);
        reponse.end('allo visiteur');
}
var serveur = http.createServer(repondeur);
serveur.listen(3000);
```

- Lancer le service

C:\>cd nodejs

C:\nodejs>

nodejs **allo-clients**

_

- Autoriser le firewall
  ufw allow 3000/tcp
- Tester avec le fureteur à
  http://localhost:3000/

https://nodejs.org/api/https.html et https://nodejs.org/api/http.html

Programme 'hello world' en tant que serveur

Fichier allo-clients.js

```
var http = require('http');
var repondeur = function(requete,reponse){

        reponse.writeHead(200);
        reponse.write('allo visiteur');
        reponse.end();
}
var serveur = http.createServer(repondeur);
serveur.listen(3000);
```

- Lancer le service

C:\>cd nodejs

C:\nodejs>

nodejs **allo-clients**

_

- Autoriser le firewall
  ufw allow 3000/tcp
- Tester avec le fureteur à
  http://localhost:3000/

https://nodejs.org/api/https.html et https://nodejs.org/api/http.html



Débugger les applications node.js: trace ou breakpoint

Utiliser console.log( )

var para = {truc: 'blabla'};

console.log(para);

ou

var para = {truc: 'blabla'};

console.log('Allo %s, mon objet: %j', 'Monde', para);

Utiliser le debugger

> `node debug monfichier.js`

WebKit Inspector

(À venir)



Séparer des fichiers





Programme avec une classe externe

Fichier main.js

```
var truc = require('./Truc');
truc.operer();
```

Fichier Truc.js

```
exports.operer = function()
{
  console.log('Allo !');
}
```

Invocation

cd .. (répété)

```
C:\>cd nodejs
C:\nodejs>node main
```

require() dans quel ordre ?

Module Core

Node.js regarde si il existe un module 'core' portant ce nom. (Des includes paths à venir ?)

1

Module dans le répertoire

Node.js explore le répertoire de déploiement si un module contribué existe tel que lemodule/index.js

2

Module en fichier dans le répertoire

Nodejs vérifie si un fichier de module existe tel que lemodule.js

3

Fonction Callback

FICHIER service.js

```
var moutonDAO = require('./MoutonDAO');
var http = require('http');

var repondre = async function(requete, reponse)
{
        var listeMoutons = await moutonDAO.listerMoutons();
        reponse.end(JSON.stringify(listeMoutons));
}

var serveur = http.createServer(repondre);
serveur.listen(8080);
```



PATTERN pour EVENT





> Node.js fonctionne selon le design pattern observer avec les EventEmitters



OBSERVER

Ce design pattern signifie que des combinaisons d'objets et d'événements sont 'écoutés' pour déclencher des réponses adaptées.

Programmer avec les EventEmitters

```
var http = require('http');
var serveur = http.createServer(
function(requete, reponse)
{


});
serveur.listen(8080);
```

```
var data = '';
requete.on('data', function(message) {
   data += message;
 }).on('end', function() {
   console.log('POST data: %s', data);
 });
reponse.end('Donnees recues');
```

https://nodejs.org/api/https.html et https://nodejs.org/api/http.html

Serveur echo

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

http.createServer(function(requete, reponse) {
  requete.on('error', function(err) {
    console.error(err);
    reponse.statusCode = 400;
    reponse.end();
  });
  reponse.on('error', function(err) {
    console.error(err);
  });
  if (requete.method === 'GET' && requete.url === '/echo') {
    request.pipe(response);
  } else {
    reponse.statusCode = 404;
    reponse.end();
  }
}).listen(8080);
```

https://nodejs.org/en/docs/guides/anatomy-of-an-http-transaction/

Combinaison des trois: requete, body, réponse

```
if (requete.method === 'GET' && requete.url === '/echo')
{
   var body = [];
   request.on('data', function(chunk)
   {
       body.push(chunk);
   })
   .on('end', function()
   {
       body = Buffer.concat(bod
       response.end(body
   })
 }
else
{
   response.statusCode = 404;
   response.end();
 }
```

On arrête ici pour le cours Multi-joueur

https://nodejs.org/en/docs/guides/anatomy-of-an-http-transaction/

Neuf comme vieux

Avec l'interface **Submittable**

https://node-postgres.com/guides/upgrading

```
const { Client, Query } = require('pg')
const query = client.query(ne
query.on('ro

})
quer

})
query

})
```

On arrête ici pour le cours Multi-joueur



Autre





Merci!

Des questions?

Vous me trouvez:

- @nadineducegep
- nadineducegep@gmail.com





Crédits

Un grand merci à ces contributeurs pour leurs merveilleuses ressources gratuites:

- Mise en page par SlidesCarnival
- Images par Unsplash
- Inspiration par http://nodeguide.com/beginner.html

