ÉCOLE POLYTECHNIQUE DE LOUVAIN

LINFO1212

Projet d’approfondissement en informatique

NodeJS / Express

Siegfried Nijssen

14 octobre 2019

Le système de modules node.js

Node.JS

Node.js est un environnement permettant d'exécuter du JavaScript sur un serveur au lieu dans le navigateur web

→ On va utiliser Node.JS pour créer un serveur web

Créé en 2009 en utilisant le moteur V8 de Google.

Permet d’exécuter JavaScript sur la ligne de commande

express: Serveur web pour node.js

Express est un module pour Node.JS qui permet de créer des serveurs web

demo-express-static.js

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

app.use(express.static('content'));
app.listen(8080);
```

Modules node.js

- node.js propose un système de modules simple
- 1 fichier = 1 module, comme en Python
- function, class, const, let sont locales au module
- Chaque module exporte (rend visible) une valeur via module.exports

circle.js

```
// fonction locale au module
function piRadius(r) { return Math.PI * r }

module.exports = {          // le module exporte un objet
   area: function(r) { return r * piRadius(r) },
   circumference: function(r) { return 2 * piRadius(r) }
};
```

Importation

Un module peut importer un autre module via require()

app.js

```
let circle = require('./circle.js');
console.log(`Area of circle of radius 4 = ${circle.area(4)}`
```

circle.js

```
// fonction locale au module
function piRadius(r) { return Math.PI * r }

module.exports = {          // le module exporte un objet
   area: function(r) { return r * piRadius(r) },
   circumference: function(r) { return 2 * piRadius(r) }
};
```

require() renvoie la propriété exports du module comme résultat

Types de modules

Modules du noyaux

Compilés dans le binaire (node)

```
let path = require('path'),
    http = require('http'),
    fs = require('fs')
```

Modules internes

Chemin relatif au fichier courant

```
let circle = require('./circle'),
    square = require('./square')
```

Modules externes

Chargés à partir de node_modules/

```
let express = require('express'),
    _ = require('underscore')
```

npm : gestionnaire de modules pour node.js

Gestionnaire du répertoire node_modules,
contenant les modules installés

Installation d'un module

```
> npm install underscore
```

Stocke le module dans le dossier `node_modules`

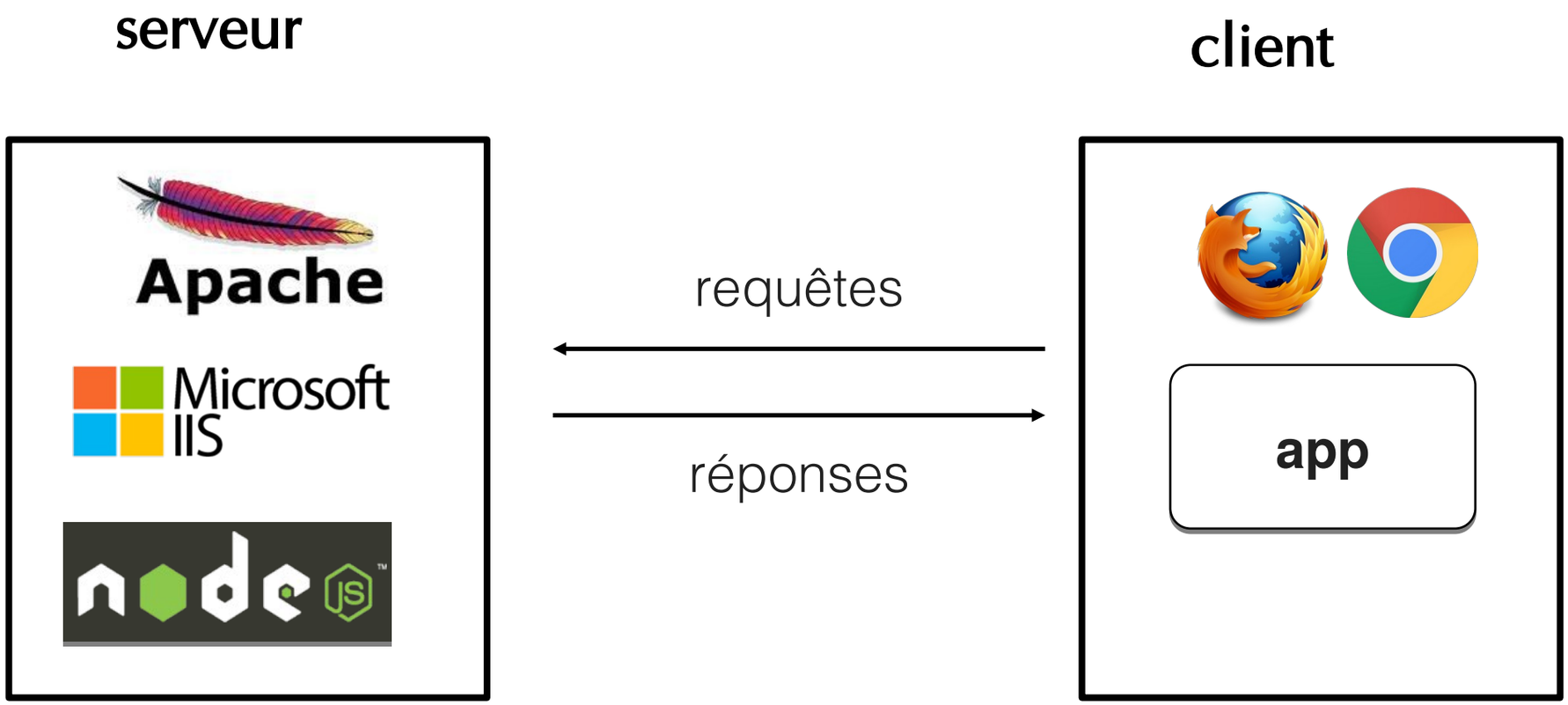
express: Serveur web pour node.js

Express est un module qui permet de créer des serveurs web

demo-express-static.js

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

app.use(express.static('content'));
app.listen(8080);
```

Architecture client-serveur



- Les clients ne sont pas nécessairement des navigateurs web, mais le seront dans notre projet

Le protocole HTTP

- Un protocole lisible (en ASCII) pour l'échange d'information
- Une **requête** du client se compose de :
  - une *méthode* de communication (GET, POST, …)
  - un identificateur de *resource* (URI)
  - version du *protocole* supporté par le client
  - des *entêtes* concernant la requête
  - *Optionnel: corps* (« contenu ») de la requête

GET /page.html HTTP/1.1
Host: example.org
User-Agent: Mozilla/5.0
Accept: text/html,*/*
Accept-Language: en-US

POST /handler HTTP/1.1
Host: example.org
…
Content-Type: application/json
Content-Length: 19

{ "user" : "Jhon" }

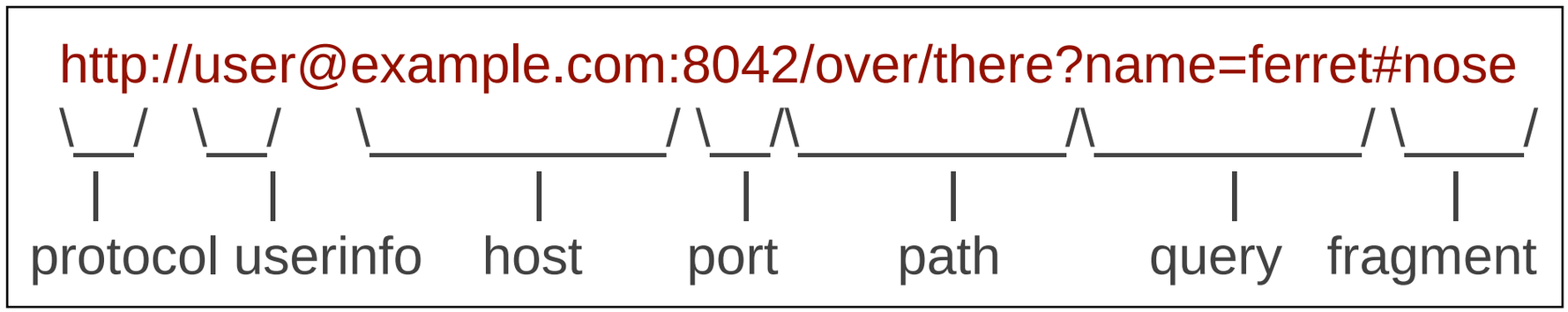
PUT /file.txt HTTP/1.1
Host: example.org
…
Content-Type: text/plain
Content-Length: 93

Lorem ipsum dolor sit amet …

DELETE /image.jpg HTTP/1.1
Host: example.org

Uniform Resource Identifiers (URI)

RFC3986



- Le « query » incorpore des donnés pour la requête directement dans l'URL.

https://google.com/search?q=express

Query : q=express

- Le query peut contenir des paires param=value separés par &.

https://youtube.com/watch?v=ldPf3yqq3-8&list=RDldPf3yqq3-8

Query : v=ldPf3yqq3-8&list=RDldPf3yqq3-8

Paires : v=ldPf3yqq3-8, list=RDldPf3yqq3-8

Le protocole HTTP

- Use **réponse** du serveur de :
  - la version du *protocole*
  - code du *résultat*
    - 2xx = succes
    - 3xx = ressource déplacée
    - 4xx = problème d'accès
    - 5xx = erreur interne
  - *entêtes* concernant la réponse
  - *corps* (« contenu ») de la réponse

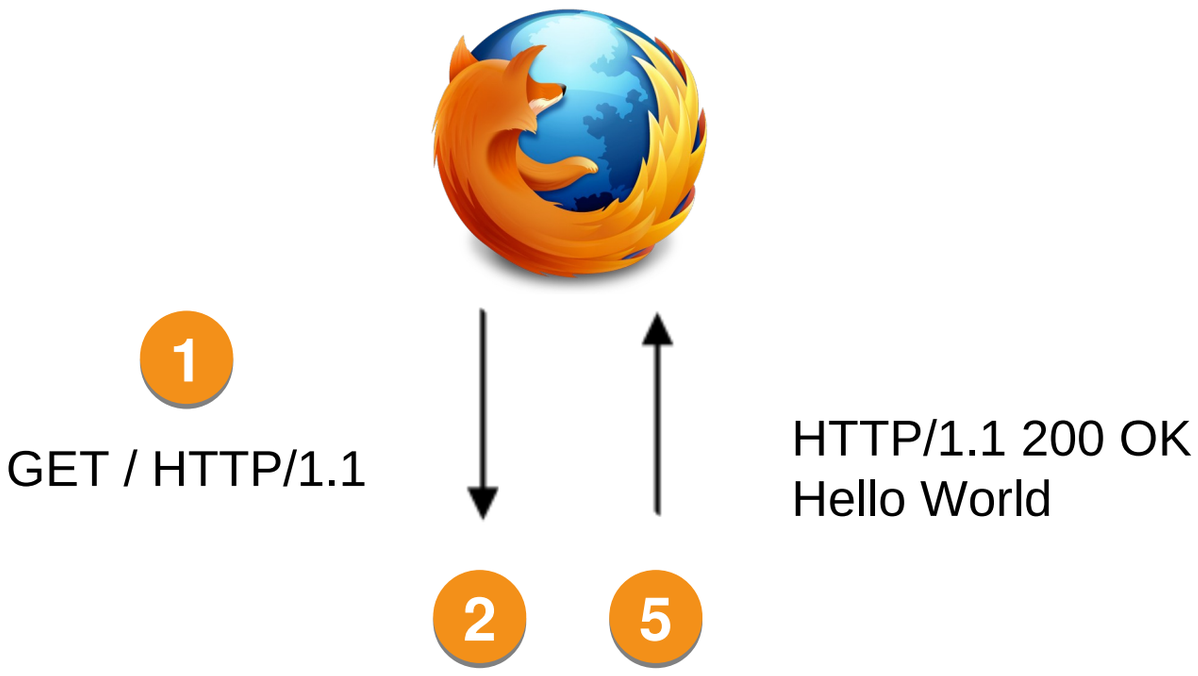
```
HTTP/1.1 200 OK
Server: Apache/0.8.4
Content-Type: text/html
Content-Length: 105

<!DOCTYPE html>
<html>
  <head>
  ...
```

```
HTTP/1.1 201 Created
Content-Length: 0
```

```
HTTP/1.1 404 Not Found
Content-Type: text/html
Content-Length: 1635

<h1>Oops…</h1>
```

```
HTTP/1.1 500 Internal Error
Content-Type: text/plain
Content-Length: 8

Problem!
```

Interaction client–serveur HTTP en node.js



1. Le client commence une requête HTTP.
2. node.js accepte la connexion et passe les donnes de la requête au serveur express.
3. Le serveur express analyse les en-têtes de la requête et appelle le callback.
4. Le callback exécute la logique de l'application et produit une réponse.
5. Le serveur construit une réponse HTTP avec les en-têtes et contenu spécifiés.

express: Moteur web pour node.js

On peut enregistrer plusieurs callbacks, pour des chemins différents

demo-express-url.js

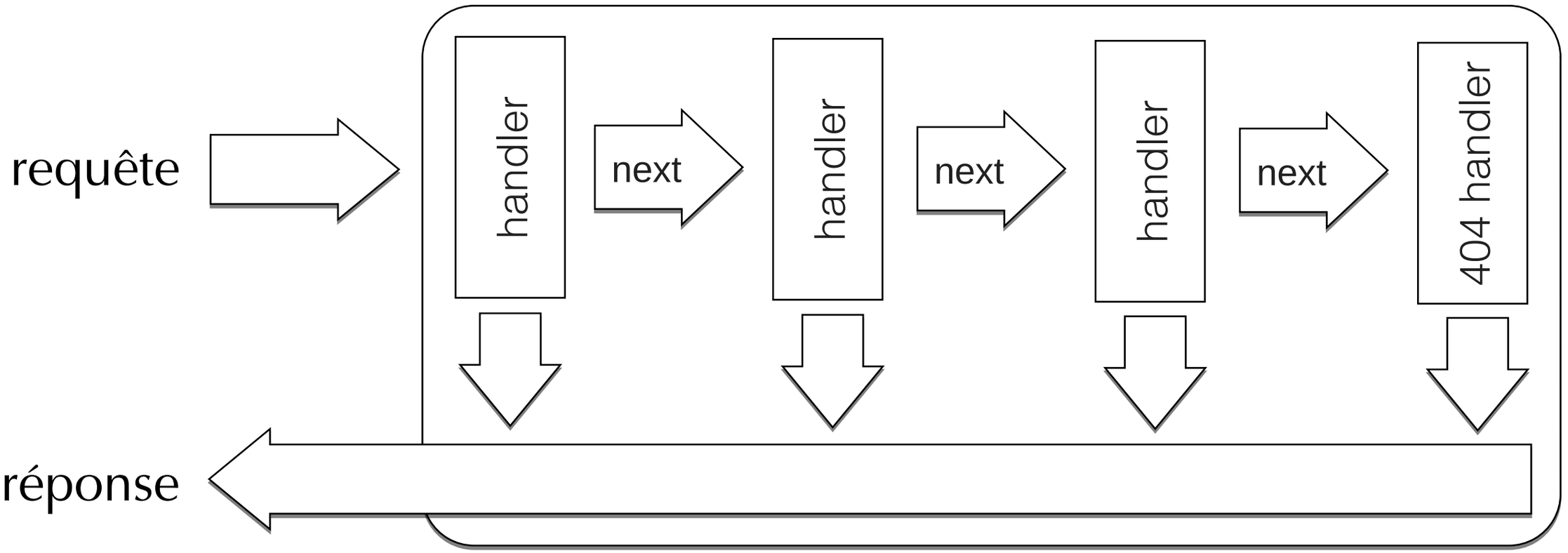
```
var express = require('express');
var app = express ();

app.get('/question', function(req,res,next) {
  res.send(`Year: ${req.query.year}` );
});
app.use(express.static('content'));
app.listen(8080);
```

Path

Réponse

Pour tous les autres cas

Traitement des requêtes: Middleware



- définition de middleware :

```
function logger(req, res, next) {
    console.log('%s %s', req.method, req.url);
    next();
};
```

- enregistrement dans la pile de middleware :

```
app.use(logger);
```

Traitement des requêtes: Middleware

Exemple pour montrer sur la console tous les URLs demandés

demo-express-url-log.js

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

app.use(function(req,res,next) {
  console.log ( req.url );
  next();
});
app.get('/question', function(req,res,next) {
  res.send(`Year: ${req.query.year}` );
});
app.use(express.static('content'));
app.listen(8080);
```

Traitement des requêtes: query

- Façon d'encoder des donnés directement dans l'URL d'une requête.

- GET /search?q=tobi+ferret
  GET /search?q=tobi%20ferret
  → req.query.q === “tobi ferret”

- GET /print?color&size=A4&orientation=landscape
  → req.query.color === “”
  → req.query.size === “A4”
  → req.query.orientation === “landscape”

- GET /shoes?order=desc&shoe[color]=blue&shoe[type]=converse
  → req.query.order === "desc"
  → req.query.shoe.color === "blue"
  → req.query.shoe.type === "converse"

Traitement des réponses: Vues

Comment est-ce qu'on peut retourner un fichier HTML élaboré?

Pas pratique:

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

app.get('/question', function(req,res,next) {
  res.send(`
      <html>
        <head>
        <title>Reponse</title>
        </head>
        <body>
           Year: ${req.query.year}
        </body>
      </html>` );
});
app.use(express.static('content'));
app.listen(8080);
```

Traitement des réponses: Vues

Comment est-ce qu'on peut retourner un fichier HTML élaboré?

demo-express-hogan.js

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

app.engine ( 'html', consolidate.hogan )
app.set('views', 'private');

app.get('/question', function(req,res,next) {
  res.render('year.html', {year: req.query.year} );
});
app.use(express.static('content'));
app.listen(8080);
```

Traitement des réponses: Vues

Comment est-ce qu'on peut retourner un fichier HTML élaboré?

private/year.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Year</title>
    <link rel="stylesheet" type="text/css" href="mystyle.css">
  </head>
  <body>
     <h1>Year</h1>
     <p>
        The year indicated is:
        <p>
           <h2>{{year}}</h2>
        </p>
     </p>
  </body>
</html>
```

Sera remplacé

Création de requêtes: formulaires

- Le formulaire définit la méthode HTTP utilisée pour envoyer les données

index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Year</title>
    <link rel="stylesheet" type="text/css" href="mystyle.css">
  </head>
  <body>
      <h1>Name Application</h1>
      <p>
          Enter the following:
          <form action="display" method="get">
          First name:<br>
          <input type="text" name="firstname"><br>
          Last name:<br>
          <input type="text" name="lastname">
          <input type="submit" value="Show">
          </form>
      </p>
  </body>
</html>
```

Création de requêtes: formulaires

- Pour réagir:

demo-express-forms.js

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

app.engine ( 'html', consolidate.hogan )
app.set('views', 'private');


app.get('/display', function(req,res,next) {
  res.render('name.html',
      {firstname: req.query.firstname,
        lastname: req.query.lastname} );
});
app.use(express.static('nameapp'));
app.listen(8080);
```

Autre Middleware

- Cookies: stocker des informations sur l'ordinatieur de l'utilisateur
- HTTPS: connection secure
- Mot de passe: OpenID (Google), Oauth (Facebook), local