

Formation Node.js

Romain Bohdanowicz

Twitter: @bioub - https://github.com/bioub

http://formation.tech/



Node.js

Node.js - Introduction



- Créé 2009 par Ryan Dahl
 A l'origine, Ryan Dahl voulait simplifier la création d'une barre d'upload.
- Projet de la OpenJS Foundation
- Un programme en ligne de commande combinant :
 - le moteur JavaScript V8 de Chrome
 - une boucle d'événement
 - une gestion bas niveau des entrées/sorties
 - des libs externes (openssl, zlib...)

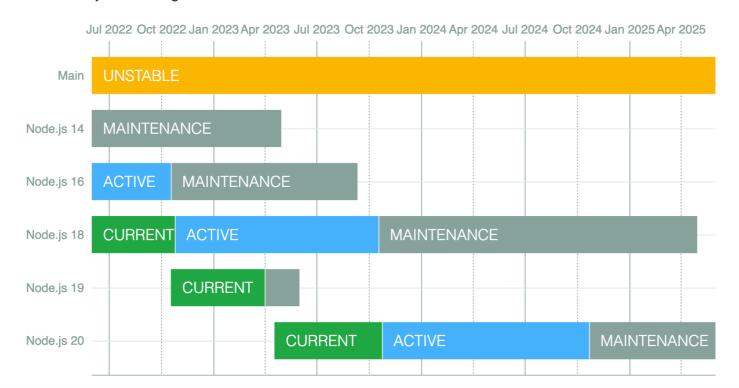
Node.js - LTS



Release schedule

Release	Status	Codename	Initial Release	Active LTS Start	Maintenance Start	End-of- life
14.x	Maintenance	Fermium	2020-04-21	2020-10-27	2021-10-19	2023-04- 30
16.x	Active LTS	Gallium	2021-04-20	2021-10-26	2022-10-18	2023-09- 11
18.x	Current		2022-04-19	2022-10-25	2023-10-18	2025-04- 30
19.x	Pending		2022-10-18	-	2023-04-01	2023-06- 01
20.x	Pending		2023-04-18	2023-10-24	2024-10-22	2026-04- 30

Dates are subject to change.



Node.js - Installation



Windows

Exécutables: https://nodejs.org/download/

nvs:

https://github.com/jasongin/nvs

OS X

Exécutables: https://nodejs.org/download/

Ou via homebrew: brew install node

- Debian / Ubuntu
 sudo apt-get update
 sudo apt-get install nodejs npm
- Pensez à ajouter le répertoire de Node au Path.
- OSX / Linux https://github.com/nvm-sh/nvm

•

Node.js - Helloworld



Lancement du programme node FILE_PATH[.js]

```
/* Un simple helloword */
/** @function helloworld */
function helloworld() {
     'use strict'; // bonne pratique
    console.log('Helloworld');
setInterval(helloworld, 1000);
                                                                  LearningJS - node - 78×16
                                       MacBook-Pro-de-Romain:LearningJS romain$ node Node.js/Slides/helloworld.js
                                       Helloworld
                                       Helloworld
                                       Helloworld
                                       Helloworld
                                       Helloworld
                                       Helloworld
                                       Helloworld
                                       Helloworld
                                       Helloworld
```

Node.js - Débogage



- Avec les DevTools de Chrome
 node --inspect-brk FILE_PATH[.js]
 Puis sous Chrome aller à l'URL <u>chrome://inspect</u>
- Avec les IDEs Webstorm ou Visual Studio Code

Node.js - 2 utilisations



- Pour les programmes en ligne de commande, ex :
 - build: webpack / grunt / gulp...
 - linters: eslint / tslint / stylelint...
 - tests: mocha / jest / karma
 - serveurs web de dev: http-server / live-server...
 - •
- Pour les programmes serveur
 - · Applications traditionnelles, HTML rendu côté serveur
 - API REST
 - Serveur WebSocket
 - •

Node.js - Pourquoi JavaScript côté serveur?



```
const http = require('http');
http.createServer((req, res) => {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello, world !');
}).listen(8080);
```

- Avantage du JavaScript côté serveur
 - Performance : le côté non-bloquant de JavaScript le rend particulièrement performant, plus besoin de gérer les problèmes inter-thread ou inter-processus
 - Réutilisation : une bibliothèque ou un composant peut être utilisé sur le client comme sur le serveur
 - Apprentissage: vous connaissez déjà JavaScript
 - Ecosystème : le nombre de bibliothèques open-source (langage le plus populaire sur GitHub)

Node.js - Event Loop



```
const http = require('http');
http.createServer((req, res) => {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello, world !');
}).listen(8080);
```

- Node.js implémente côté C++ une boucle d'événement et une gestion non-bloquante des entrées/sorties
- Ici lorsque qu'un requête HTTP arrive sur le port 8080 la fonction de rappel passée en argument de createServer est appelée
- Il faut quelques millisecondes pour exécuter ce callback, le reste du temps JavaScript est libre de faire autre chose (exécuter des requêtes concurrentes, se connecter à une base de données, écrire dans un fichier...)

Node.js - Documentation



- Livre https://nodejs.dev/
- Guides https://nodejs.org/en/docs/guides/
- APIhttps://nodejs.org/api/
- 3 niveaux de "stabilité"

Stability: 0 - Deprecated. This feature is known to be problematic, and changes may be planned. Do not rely on it. Use of the feature may cause warnings to be emitted. Backwards compatibility across major versions should not be expected.

Stability: 1 - Experimental. This feature is still under active development and subject to non-backwards compatible changes, or even removal, in any future version. Use of the feature is not recommended in production environments. Experimental features are not subject to the Node.js Semantic Versioning model.

Stability: 2 - Stable. The API has proven satisfactory. Compatibility with the npm ecosystem is a high priority, and will not be broken unless absolutely necessary.

Node.js - Chapitres de la Doc API



•	Assertion	Testing

- Async Hooks
- Buffer
- C++ Addons
- C/C++ Addons N-
- Child Processes
- Cluster
- Command LineOptions
- Console
- Crypto
- Debugger

- Deprecated APIs
- DNS
- Domain
- ECMAScript Modules
- Errors
- Events
- File System
- Globals
- HTTP
- HTTP/2
- HTTPS
- Inspector

- Internationalizatio
- Modules
- Net

n

- · OS
- Path
- Performance Hooks
- Process
- Punycode
- Query Strings
- Readline
- → REPL

- Stream
- String Decoder
- Timers
- TLS/SSL
- Trace Events
- TTY
- UDP/Datagram
- URL
- Utilities
- V8
- VM
- Worker Threads
- ZLIB

Node.js - API



- Ne sont pas (ou pas complètement) des APIs
 - C++ Addons et C/C++ Addons N-API
 Permet de compiler du code C ou C++ et de le "binder" (le rendre accessible) au JavaScript de Node.js
 - Command Line Options
 Arguments du programme Node
 - Debugger
 Un debugger en ligne de commande
 - Deprecated APIs
 La liste des APIs dépréciés
 - ECMAScript Modules
 L'implémentation des Modules ECMAScript (ESM ou ES6 Modules)
 - Internationalization
 Comment compiler Node.js pour améliorer le support i18n
 - REPL
 Permet de lancer Node en mode interactif et de saisir du code dans la console

Node.js - API



- Les APIs dépréciés (seront supprimé à l'avenir)
 - Domain
 - Punycode
 - https://nodejs.org/dist/latest/docs/api/deprecations.html
- Les APIs expérimentaux (leur implémentation peut évoluer)
 - Async Hooks
 - Inspector (jusqu'à Node 12)
 - Trace Events

Node.js - Console



Le module console (global) permet de logger dans la console et de réaliser des benchmarks

```
console.time('100-elements');
for (var i = 0; i < 100; i++) {
    console.log(i);
}
console.timeEnd('100-elements');
// 100-elements: 8ms</pre>
```

Node.js - Timers



Le module Timers (global) contient les fonctions pour différer l'exécution de callbacks.

```
setTimeout(function() {
    console.log('1 fois dans 3 secondes');
}, 3000);

var intervalId = setInterval(function() {
    console.log('toutes les 2 secondes');
}, 2000);

setTimeout(function() {
    console.log('Bye bye');
    clearInterval(intervalId);
}, 15000);
```

Node.js - File System



Le module File System (require(fs)) permet les accès aux disques, fichiers, dossiers, droits, etc...

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

try {
    var stats = fs.statSync('./dist');
}
catch(e) {
    fs.mkdirSync('./dist');
}

var fichiers = fs.readdirSync('./src');

for (var i=0; i<fichiers.length; i++) {
    var fichier = fichiers[i];
    var src = './src/' + fichier;
    var dest = './dist/' + fichier;

    var contenu = fs.readFileSync(src);
    fs.writeFileSync(dest, contenu);
}</pre>
```

Node.js - Net



Le module net (require(net)) permet les accès réseau

```
var net = require('net');
var server = net.createServer(function(c) { //'connection' listener
        console.log('client connected');
        c.on('end', function() {
            console.log('client disconnected');
        });
        c.write('hello\r\n');
        c.pipe(c);
});
server.listen(8124, function() { //'listening' listener
        console.log('server bound');
});
```

Node.js - Serveur Net



Un chat serveur avec Net

```
var net = require('net');
var clients = {}, cpt = 0;
var server = net.createServer(function(c) { //'connection' listener
   var me = 'c' + (++cpt);
    console.log('client connected');
   clients[me] = c;
    c.on('end', function() {
       //clients[me].end();
        delete clients[me];
   });
    c.write('Bienvenue sur le Chat !!! (telnet : taper exit pour quitter)\r\n');
    c.on('data', function(chunk) {
        for (var cid in clients) {
            if (clients.hasOwnProperty(cid)) {
                if (chunk.toString().indexOf('exit') === 0) {
                    clients[me].end();
                    delete clients[me];
                    break:
                if (cid !== me) {
                    clients[cid].write(chunk.toString());
        }
   })
});
server.listen(8124, function() { //'listening' listener
    console.log('server bound');
});
```

Node.js - Client Net



Un chat client avec Net

```
var net = require('net');
var readline = require('readline');
var rl = readline.createInterface({
    input: process stdin,
    output: process stdout
});
rl.question("Quel est ton pseudo ? ", function(pseudo) {
    console.log("Bienvenue sur le chat", pseudo);
    var client = net.connect({port: 8124}, function() { //'connect' listener
            console.log('(connecté au serveur)');
            process.stdin.on('readable', function() {
                var chunk = process.stdin.read();
                if (chunk !== null) {
                    var msg = chunk.toString();
                    msg = msg.substr(0, msg.length - 1); // on retire le \n
                    client.write(pseudo + ': ' + msg);
           });
       });
    client.on('data', function(data) {
        console.log(data.toString());
        //client.end();
    client.on('end', function() {
        console.log('disconnected from server');
    });
    rl.close();
});
```

Node.js - HTTP



- CreateServer
 Contrairement à d'autres technologies, l'implémentation du serveur HTTP se fait dans l'application.
- Callback
 Une fonction de rappel est associée au serveur. Elle sera appelée à chaque requête HTTP.
- Objets Request et Response
 Node.js abstrait la requête (IncomingMessage) et la réponse (ServerResponse), le callback doit créer une réponse valide avant la fin de son exécution.

```
var http = require('http');
http.createServer(function(req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello, world !');
}).listen(8080);
```

Node.js - HTTPS



HTTPS

Le serveur HTTPS démarre avec une clé privée et un certificat. Les serveurs HTTP et HTTPS cohabitent dans la même application.

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

function serveur(req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello, world !');
}

var options = {
    key: fs.readFileSync('./key.pem', 'utf8'),
    cert: fs.readFileSync('./server.crt', 'utf8')
};

http.createServer(serveur).listen(80);
https.createServer(options, serveur).listen(443);
```

Node.js - Gestion des routes



- Pages vs Serveur Node.js possède une implémentation HTTP très bas niveau. Quand en Java ou en PHP un fichier équivaut à une URL, Node.js lui est le serveur dans son ensemble.
- Routes
 La gestion des URL se fait donc en internes, l'association entre un chemin d'accès (Méthode HTTP + URL) et du code s'appelle une route.
- Router
 L'ensemble des routes est configurée dans un composant que l'ont appelle un router.

Node.js - Gestion des routes



Implémentation d'un router basique
 Dans l'exemple ci-dessous, on implémente un router à l'aide d'un simple switch.

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  switch (req.url) {
    case '/':
      res.end('Hello World');
      break:
    case '/toto':
      res.end('Hello Toto');
      break;
    default:
      res.statusCode = 404;
      res.end('404 Not Found');
      break;
});
server_listen(8080, () => {
 console.log('Server started http://localhost:8080');
});
```

Node.js - Exercice



- Créer un serveur web contenant 2 routes : /contacts et /contacts/123
- Sur la route /contacts retourner un tableau de contact (prenom + nom)
- Sur la route /contacts/123 retourner le contact dont l'id est 123 (utiliser la fonction find du type Array)
- Remplacer le 123 dans la route /contacts/123 pour permette de passer un id quelconque (utiliser une expression régulière ou bien la méthode substr du type String)



Express

Express - Framework Web



- Définition d'un framework web: Ensemble de composants logiciels permettant d'architecturer un projet logiciel.
- Différences par rapport à une bibliothèque :
 Le framework ne se destine pas à une tache précise (ensemble de bibliothèques)
 Le framework instaure un cadre de travail (squelettes d'application, documentation sur l'architecture...)

Express - Frameworks web connus



- Java
 Struts (2000), Spring (2003), GWT (2006), Play (2007)...
- Ruby on Rails (2005), Sinatra (2007)...
- PythonDjango (2005)...
- PHP Symfony, Zend Framework, CakePHP, CodeIgniter...

Express - Frameworks web en JS



- Clients
 AngularJS (2010), Ember.js (2011)
- Server
 Express (2009), Hapi (2012)
- Fullstack (Client + Server)Meteor (2012), Sails.js (2012)...

Express - Introduction

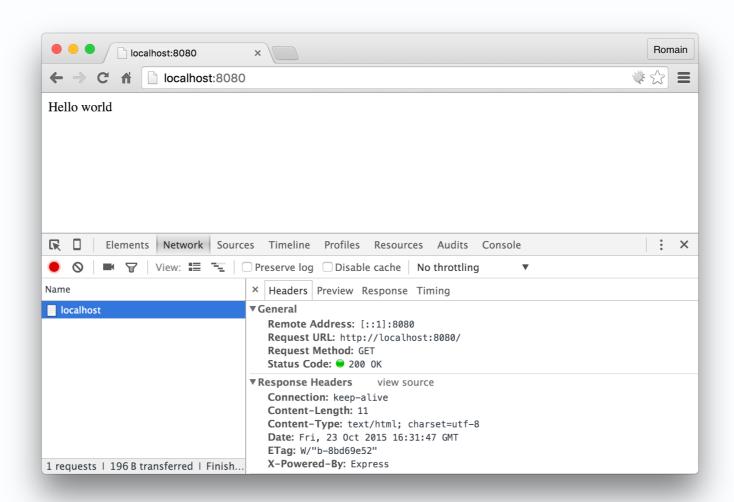


- Express
 Framework pour Node.js le plus populaire, créé en 2009, aujourd'hui en version 4.
 Permet d'architecturer plus facilement le serveur web.
 Très souvent utilisé pour construire des APIs REST.
- Avantages sur le module HTTP de Node.js
 - Gestion des URLs et des méthodes HTTP
 - Approche MVC
 - Utilisation de middlewares qui permettent d'étendre le code
 - De nombreux middleware open-source existent
 - Construit comme une surcouche de HTTP, les objets Request et Response sont simplement étendus
- Installation
 npm install express --save

Express - Helloworld



```
const express = require('express');
const app = express();
app.get('/', (req, res) => {
   res.send('Hello world');
});
app.listen(8080);
console.log("URL http://localhost:8080/");
```



Express - MVC

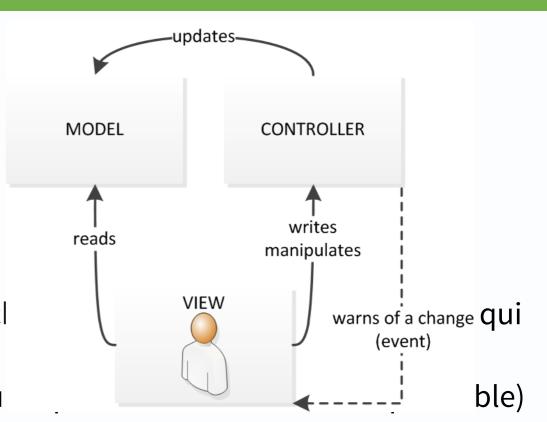


- Définition
 L'architecture MVC est un Design Pattern apparu en Smalltalk et très répandu dans les frameworks web
- Objectif
 L'objectif est de séparer les responsabilités de 3 types de composants : le Modèle (Model), la Vue (View), le Contrôleur (Controller)
- Documentation: http://martinfowler.com/eaaCatalog/modelViewController.html http://martinfowler.com/eaaDev/uiArchs.html http://fr.wikipedia.org/wiki/Modèle-vue-contrôleur

Express - MVC



- Modèle
 Données, accès aux données, validation
- Vue Rendu. Se limiter à :
 - affichage de variable
 - bloc conditionnels if .. else if .. else (ex : afficl dépend d'une authentification)
 - boucles foreach (uniquement foreach, ce qu
 - appel à des fonctions de filtrage, de formatage, de rendu (parfois appelées aides de vues)
- Contrôleur
 Analyse de la requête, interrogation du Modèle, transmission des données à la vue, gestion des erreurs, des redirections...



Express - Middleware



Définition

Un middleware est une fonction qui va s'exécuter en amont ou en aval d'une requête dans Express pour l'étendre simplement.

Exemple

Logs de requêtes, authentification, gestion des requêtes Cross-Domaines, support d'un moteur de templates...

Connect

Historiquement Express utilisait le module npm connect pour la mise en place de middleware. A partir d'Express 4, les développeurs d'Express ont développé leur propre système de middleware tout en gardant la compatibilité avec Connect.

Express - Express Generator



- Introduction
 Express fourni un générateur de squelette d'application pour démarrer rapidement ses projets web (plutôt adapté aux rendus HTML)
- Installation
 npm install -g express-generator
- Création du squelette d'application express Helloworld
- Installation des dépendances
 cd Helloworld && npm install
- Lancement de l'application
 DEBUG=HelloworldEJS:* npm start

Express - Express Generator



Autres options d'installation du squelette

```
Helloworld — -bash — 106×16
MacBook-Pro-de-Romain: Helloworld romain  express -h
  Usage: express [options] [dir]
  Options:
    -h, --help
                        output usage information
    -V, --version
                       output the version number
    -е, --еjs
                        add ejs engine support (defaults to jade)
        --hbs
                       add handlebars engine support
                        add hogan.js engine support
    -H, --hogan
    -c, --css <engine> add stylesheet <engine> support (less|stylus|compass|sass) (defaults to plain css)
                        add .gitignore
        --git
    -f, --force
                       force on non-empty directory
MacBook-Pro-de-Romain:Helloworld romain$
```

- Choix du moteur de templates
 Jade, EJS, Handlebars, Hogan.js
- Choix d'un préprocesseur CSS
 CSS, Less, Stylus, Compass, Sass



- app.js
 Configuration de l'application, objet principal
- bin/wwwDémarrage du serveur
- package.jsonDépendances npm
- public
 Fichiers statiques (images, scripts client, css, pdf...)
- routes
 Contrôleurs, configuration des URLs
- views
 Fichiers de rendus (ici au format Jade)

```
app.js
node modules
package.json
    images
    javascripts
    stylesheets
      — style.css
routes
    index.js
    users.js
views
   – error.jade
   - index.jade
    layout.jade
```



- bin/www
 - Dépendances
 - Définition du port (variable d'env PORT ou 3000)
 - Création du serveur
 - Démarrage du serveur
 - Listeners sur erreurs et démarrage

```
#!/usr/bin/env node
/**
* Module dependencies.
var app = require('../app');
var debug = require('debug')('Helloworld:server');
var http = require('http');
/**
* Get port from environment and store in Express.
 */
var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);
/**
* Create HTTP server.
 */
var server = http.createServer(app);
/**
* Listen on provided port, on all network interfaces.
*/
server.listen(port);
server.on('error', onError);
server.on('listening', onListening);
// ...
```



- app.js
 - Dépendances
 - Chargement des routes
 - Création de l'objet app
 - Définition du moteur de templates
 - Définition des middlewares à appeler avant le contrôleur
 - Définition des contrôleurs sur un préfixe d'URL
 - Middleware pour les erreurs 404
 - Middleware pour afficher les erreurs (avec env de dev et de prod)

```
var express = require('express');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var routes = require('./routes/index');
var users = require('./routes/users');
var app = express();
// view engine setup
app.set('views', path.join( dirname, 'views'));
app.set('view engine', 'jade');
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join( dirname, 'public')));
app.use('/', routes);
app.use('/users', users);
// catch 404 and forward to error handler
app.use(function(req, res, next) {
 var err = new Error('Not Found');
  err.status = 404;
 next(err);
// error handlers
```



- routes/index.js
 - Dépendances
 - Association d'un contrôleur à la l'URL GET /
 - Appel de la vues en transmettant la variable title (contenu 'Express')

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next))
{
   res.render('index', { title:
'Express' });
});

module.exports = router;
```



- views/index.jade
 - Jade: Syntaxe très concise, l'indentation fait l'imbrication des balises, parenthèses pour les attributs
 - Héritage du layout
 - Remplacement du block content du layout par celui de la vue (inspiré de Django Template Engine, Twig…)
 - Création de la balise h1 ayant comme contenu la variable title
 - Création de la balise p qui concatène Welcome to et la variable title

```
// views/index.jade
extends layout

block content
   h1= title
   p Welcome to #{title}
```

```
// views/layout.jade
doctype html
html
head
   title= title
   link(rel='stylesheet', href='/stylesheets/style.css')
body
   block content
```



- views/index.ejs
 - Syntaxe plus simple que Jade proche de PHP, ASP, JSP
 - <%= title %>:écriture de la variable title

```
<!DOCTYPE html>
<html>
    <head>
        <title><%= title %></title>
        link rel='stylesheet' href='/stylesheets/style.css' />
        </head>
        <body>
            <h1><%= title %></h1>
            Welcome to <%= title %>
        </body>
    </html>
```

Express - Routeur



Réponse à toutes les méthodes HTTP

```
router.all('/api/*', requireAuthentication);
```

Réponse aux requêtes sur certaines méthodes HTTP

Méthodes HTTP: get, post, put, head, delete, options, trace, copy, lock, mkcol, move, purge, propfind, proppatch, unlock, report, mkactivity, checkout, merge, msearch, notify, subscribe, unsubscribe, patch, search, connect

```
router.get('/', function(req, res){
   res.send('hello world');
});
```

Avec une RegExp

```
router.get(/^\/commits\/(\w+)(?:\.\.(\w+))?$/, function(req, res){
   var from = req.params[0];
   var to = req.params[1] || 'HEAD';
   res.send('commit range ' + from + '..' + to);
});
```

Express - Routeur



Route avec paramètres nommés

```
router.get('/:id', function(req, res, next) {
    var id = req.params.id;

    if (!model[id-1]) {
        return next();
    }

    res.json({
        data: model[id-1]
    });
});
```

Ne pas confondre avec la query string Ex : /contacts?page=1&limit=100

```
// GET /search?q=tobi+ferret
req.query.q
// => "tobi ferret"
```

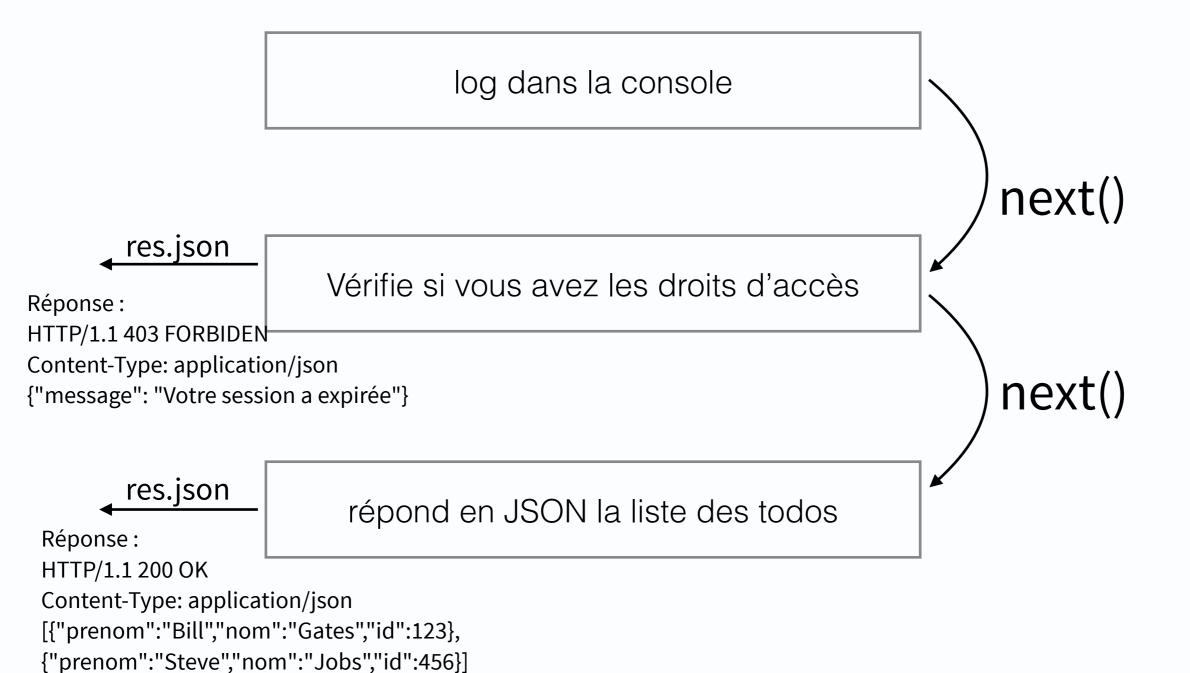


- Middleware
 Fonction qui s'exécute en amont ou en aval d'un contrôleur
- Exemple
 Ajoute les entêtes à la réponse HTTP permettant d'autoriser les requêtes Cross-Domain

```
router.use(function(req, res, next) {
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With');
    next();
});
```

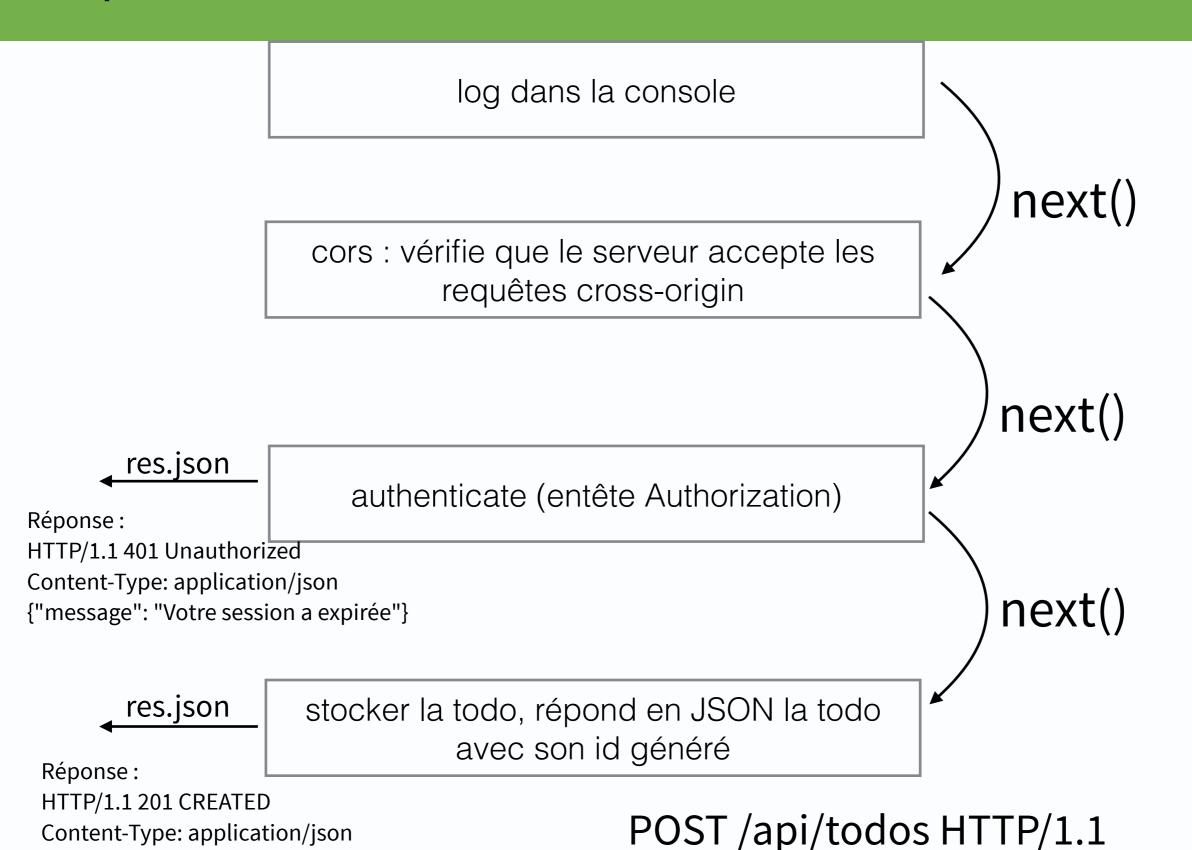


GET /api/todos HTTP/1.1



{"title": "ABC", "completed": true, id: 17}





47

{"title": "ABC", "completed": true}



- 3 middlewares
 - 1. Router qui contient toutes les URLs préfixées par /users
 - 2. Middleware appelé si l'un des contrôleurs ou middlewares précédents appelle next(), permet de gérer simplement les erreurs 404
 - 3. Middleware appelé si l'un des contrôleurs ou middlewares précédents appelle next(err) ou les erreurs non-interceptées

```
app.use('/users', users);
app.use(function(req, res, next) {
    var err = new Error('Not Found');
    err.status = 404;
    next(err);
});
app.use(function(err, req, res, next) {
    res.status(err.status || 500);
    res.render('error', {
        message: err.message,
        error: err
    });
});
```



- Request
 - L'objet Request hérite de IncomingMessage du module HTTP
- Middleware body-parser

Le middleware body-parser ajouter la propriété body à l'objet request avec le contenu du corps de requête parsé

```
var express = require('express');
var bodyParser = require('body-parser');
var app = express();
app.use(bodyParser.urlencoded({ extended: false }));
app.get('/', function(req, res) {
    var html = '<form method="post">';
       html += ' Prénom : <input name="prenom">';
       html += ' Nom : <input name="nom">';
       html += ' <button type="submit">Valider</button>';
       html += '</form>';
    res.send(html);
})
app.post('/', function(req, res) {
   // Prénom : Romain, nom : Bohdanowicz
    res.send(`Prénom : ${req.body.prenom}, nom : ${req.body.nom}`);
})
app.listen(3000);
```



Middleware multer

Le middleware multer ajouter la propriété file ou files (upload multiple) à l'objet request et contient des informations sur le fichier uploadé.

```
var express = require('express');
var multer = require('multer');
var app = express();
var upload = multer({ dest: 'uploads/' });
app.get('/', function(req, res) {
   var html = '<form method="post" enctype="multipart/form-data">';
        html += ' Fichier : <input type="file" name="fichier">';
        html += ' <button type="submit">Valider</button>';
        html += '</form>';
    res.send(html);
});
app.post('/', upload.single('fichier'), function(req, res){
    console.log(req.file);
                           { fieldname: 'fichier',
    res.status(204).end();
                             originalname: '2010_Q3.pdf',
});
                             encoding: '7bit',
                             mimetype: 'application/pdf',
app.listen(3000);
                             destination: 'uploads/',
                             filename: '799e08c05ef96ac6ec6ac5b714941161',
                             path: 'uploads/799e08c05ef96ac6ec6ac5b714941161',
```

size: 80108 }

Express - JSON



JSON

L'objet Response contient une méthode json qui sérialise un objet et renvoie les bons entêtes HTTP. Associés au méthodes de l'objet Request et aux middleware body-parser et cors, Express est le framework idéal pour la mise en place d'un API REST qui communique en JSON.

```
var app = express();
app.use(cors({ allowedHeaders: 'X-Requested-With' }));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use('/api/v1/contacts', apiContacts);
app.listen(80);
```

Express - API REST



```
var express = require('express');
var contacts = require('../model/contacts').slice(0);
var api = express.Router();
api.get('/', function(req, res) {
    res.json({contacts});
});
api.get('/:id', function(req, res, next) {
    var id = parseInt(req.params.id);
    var contact = contacts.find(elt => elt.id === id);
    if (!contact) return next();
    res.json({contact});
});
api.post('/', function(req, res, next) {
    var contact = req.body;
    contact.id = contacts[contacts.length-1].id + 1;
    contacts.push(contact);
    res.status(201);
    res.json(contact);
});
module.exports = api;
```

Express - Designer un API REST



- Guide inspiré de Google, Facebook, Twitter, GitHub... http://blog.octo.com/designer-une-api-rest/
- Guide inspiré par Heroku
 https://github.com/interagent/http-api-design
- Richardson Maturity Model
 https://martinfowler.com/articles/richardsonMaturityModel.html

Express - Exercice



 Créer un API RESTful avec Express permettant d'interagir avec un tableau de contacts en JSON

5 routes:

- GET /api/contacts Lister les contacts
- GET /api/contacts/:id Afficher un contact
- POST /api/contacts Ajouter un contact
- PUT /api/contacts/:id Modifier un contact
- DELETE /api/contacts/:id Supprimer un contact



NoSQL

NoSQL - Introduction



NoSQL

Not Only SQL, le nom qu'on donne au mouvement depuis quelques années de ne pas tout stocker sous la forme de base de données relationnelles (MySQL, SQLite, PostgreSQL, Oracle, SQL Server...).

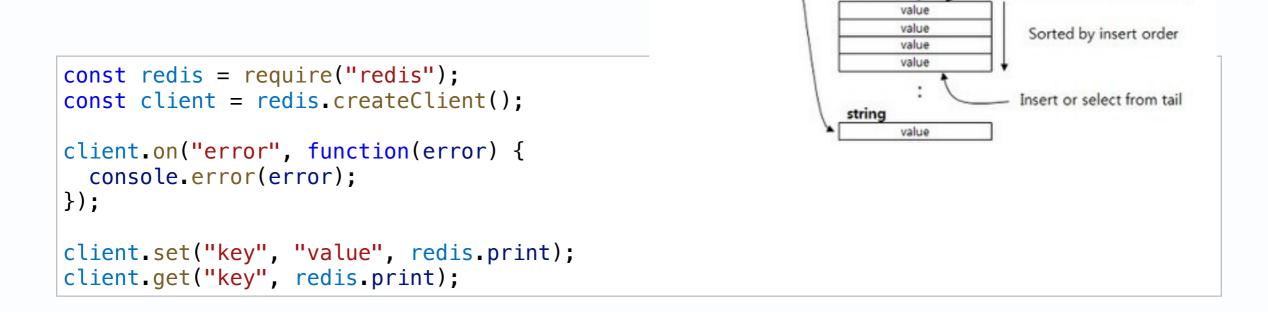
A l'origine en 2009, le nom donné à un meetup à San Francisco. Parfois appelé « NoRel » (« not only relational ») pour ne pas prêter à confusion.

- Intérêts
 Performance, scalabilité, haute-disponibilité
- Catégories
 - Clé / valeur (Redis / Memcached...)
 - Orienté Colonne (HBase / Cassandra...)
 - Orienté Document (MongoDB / CouchDB…)
 - Orienté Graphe (Neo4j)
 - ...

NoSQL - Clé / valeur



Exemple Redis



hashes

field

field

field

sorted sets

score

score

score

score

member

member

sets

F lists

value

value

value

value

member member

member

Sorted by score value

Sorted by insert order

Insert or select from top

Key

Key

Key

Key

Key

Key

NoSQL - Orienté Colonne



Exemple Cassandra

```
const cassandra = require('cassandra-driver');

const client = new cassandra.Client({
   contactPoints: ['h1', 'h2'],
   localDataCenter: 'datacenter1',
   keyspace: 'ks1'
});

const query = 'SELECT name, email FROM users WHERE key = ?';

client.execute(query, [ 'someone' ])
   .then(result => console.log('User with email %s', result.rows[0].email));
```

	А	В	С	D	E
1	foo	bar	hello		
2		Tom			
3			java	scala	cobol

1 A foo B bar C hello
2 B Tom
3 C java D scala E cobol

Organisation d'une table dans une BDD relationnelle Organisation d'une table dans une BDD orientée colonnes

NoSQL - Orienté Document



Exemple CouchDB

```
const cradle = require('cradle');
const db = new(cradle.Connection)().database('starwars');

db.get('vader', function (err, doc) {
    doc.name; // 'Darth Vader'
    assert.equal(doc.force, 'dark');
});

db.save('skywalker', {
    force: 'light',
    name: 'Luke Skywalker'
}, function (err, res) {
    if (err) {
        // Handle error
    } else {
        // Handle success
}
});
```

NoSQL - Orienté Graphe



Exemple neo4j

```
const neo4j = require('neo4j-driver')
const driver = neo4j.driver(uri, neo4j.auth.basic(user, password))
const session = driver.session()
const personName = 'Alice'
try {
  const result = await session.run(
    'CREATE (a:Person {name: $name}) RETURN a',
    { name: personName }
                                                      = (:Category)<--(:Place)-[*]->(:Terminal {name:'A'}) RETURN p
  const singleRecord = result.records[0]
  const node = singleRecord.get(0)
  console.log(node.properties.name)
} finally {
  await session.close()
// on application exit:
await driver.close()
```



- MongoDB
 Base de données écrite en C++, inclus le moteur JS SpiderMonkey de Mozilla ou Node.js (donc V8)
- Document MongoDB permet de manipuler des objets structurés au format BSON (JSON binaire). Les données prennent la forme de documents enregistrés eux-mêmes dans des collections.
- Accès aux données
 L'accès aux données se fait via un protocole réseau, les requêtes sont décrites sous forme d'objet JavaScript
- Absence de Schéma
 Contrairement à un SGBDR, les documents stockés dans une collection peuvent avoir des formats complètement différents. Les données peuvent également être imbriquées.

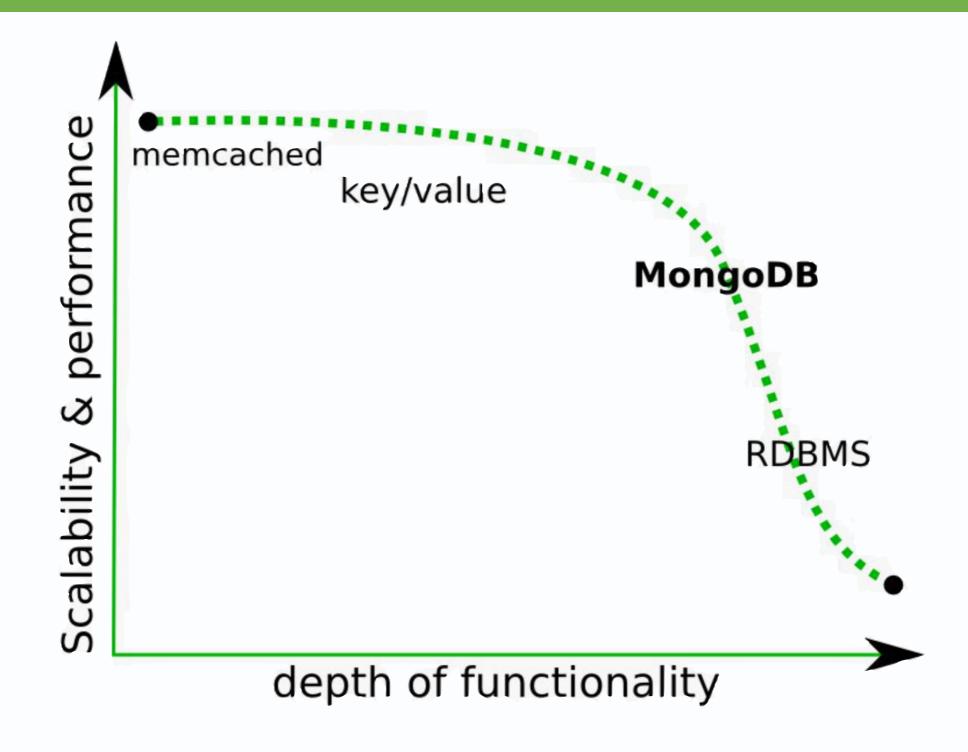


Installation

Windows
 https://www.mongodb.com/
 https://www.mongodb.org/dl/win32

Mac https://www.mongodb.com/ brew install mongo







Parallèle avec un SGBDR

MongoDB	SGBDR	
Base de données	Base de données	
Collection	Table	
Document	Enregistrement	
Pas de schéma	Schéma	
API JavaScript	SQL	



- Jeu de données d'exemple fourni par Mongo
 https://raw.githubusercontent.com/OpenKitten/Mongo-Assets/master/primer-dataset.json
- ' Importer un jeu de données
 mongoimport --collection restaurants --file ~/downloads/
 primer-dataset.json



MongoShell
 Mongo livre un programme client en ligne de commande pour accéder à la base.

```
romain — mongo — 91×17
MBP-de-Romain:∼ romain$ mongo
MongoDB shell version: 3.0.7
connecting to: test
> use address_book
switched to db address_book
> db.contact.find()
{ "_id" : ObjectId("562d4e878561c01ec2e43cfb"), "prenom" : "Steve", "nom" : "Jobs" }
{ "_id" : ObjectId("562d4e918561c01ec2e43cfc"), "prenom" : "Bill", "nom" : "Gates" } 
{ "_id" : ObjectId("562d4eab8561c01ec2e43cfd"), "prenom" : "Mark", "nom" : "Zuckerberg" }
|> db.contact.insert({prenom: 'Steve', nom: 'Ballmer'})
WriteResult({ "nInserted" : 1 })
|> db.contact.find({prenom: 'Steve'})
{ "_id" : ObjectId("562d4e878561c01ec2e43cfb"), "prenom" : "Steve", "nom" : "Jobs" }
{ "_id" : ObjectId("562d4ee1321ac0f47f03ce9d"), "prenom" : "Steve", "nom" : "Ballmer" }
>
```



Principales Commandes MongoShell

Shell Helpers	JavaScript Equivalents		
show dbs, show databases	<pre>db.adminCommand('listDatabases')</pre>		
use <db></db>	<pre>db = db.getSiblingDB('<db>')</db></pre>		
show collections	<pre>db.getCollectionNames()</pre>		
show users	db.getUsers()		
show roles	<pre>db.getRoles({showBuiltinRoles: true})</pre>		
show log <logname></logname>	<pre>db.adminCommand({ 'getLog' : '<logname>' })</logname></pre>		
show logs	<pre>db.adminCommand({ 'getLog' : '*' })</pre>		
it	<pre>cursor = db.collection.find() if (cursor.hasNext()){ cursor.next(); }</pre>		



- MongoClient
 API officiel fourni MongoDB pour accéder aux données sous Node.js
- Installation
 npm install mongodb --save
- Insertion

```
var MongoClient = require('mongodb').MongoClient;
var url = 'mongodb://localhost:27017/addressbook';
MongoClient.connect(url, function(err, db) {
    if (err) {
        console.log('Erreur : ' + err);
        return;
    }
    var cursor = db.collection('contacts').insert({prenom: 'Romain', nom: 'Bohdanowicz'},
function(err, result) {
        if (err) {
            console.log('Erreur : ' + err);
            return;
        }
        console.log('Le contact a bien été inséré');
});
```



Modification

```
var cursor = db.collection('contacts').update({nom: 'Bohdanowicz'}, {prenom: 'ROMAIN', nom:
'BOHDANOWICZ'}, {upsert:true}, function(err, result) {
    if (err) {
        console.log('Erreur: ' + err);
        return;
    }

    console.log('Le contact a bien été mis à jour');
});
```

Suppression

```
var cursor = db.collection('contacts').removeOne({nom: 'BOHDANOWICZ'}, function(err,
result) {
    if (err) {
        console.log('Erreur : ' + err);
        return;
    }

    console.log('Le contact a bien été supprimé');
});
```



Recherche

```
var MongoClient = require('mongodb').MongoClient;
var url = 'mongodb://localhost:27017/addressbook';
MongoClient.connect(url, function(err, db) {
    if (err) {
        console.log('Erreur : ' + err);
        return;
    }
    var cursor = db.collection('contacts').find();
    cursor.toArray(function(err, contacts) {
        console.log(contacts);
        db.close();
    });
});
```



MongoClient — -bash — 70×9

Recherche multi-critères
 Exemple: Restaurants de Brooklyn, ET dont la cuisine est française OU italienne ET dont l'une des notes est supérieur à 40

```
MBP-de-Romain:MongoClient romain$ node multicriteres.js
var MongoClient = require('mongodb').MongoClient;
                                                                 Nom : Doc Wine Bar, cuisine : Italian, adresse : 83 North
                                                                 Nom : Le Gamin, cuisine : French, adresse : 556 Vanderbilt Avenue
                                                                 Nom : Peperoncino, cuisine : Italian, adresse : 72 5 Avenue
var url = 'mongodb://localhost:27017/test';
                                                                 Nom : Patrizia'S, cuisine : Italian, adresse : 35 Broadway
                                                                 Nom : Tutta Pasta, cuisine : Italian, adresse : 160 7 Avenue
MongoClient.connect(url, function(err, db) {
                                                                 Nom: Anella, cuisine: Italian, adresse: 222 Franklin Street
                                                                 Nom : Joe'S Pizza, cuisine : Italian, adresse : 349 5 Avenue
    if (err) {
                                                                 MBP-de-Romain:MongoClient romain$
         console.log('Erreur : ' + err);
         return;
    var cursor = db.collection('restaurants').find({
         borough: 'Brooklyn',
         sor:
             { "cuisine": "Italian" },
             { "cuisine": "French" } ],
         'grades.score': { $qt: 40 }
    });
    cursor.toArray(function(err, restaurants) {
         restaurants.forEach(function(r) {
              console.log(`Nom : ${r.name}, cuisine : ${r.cuisine}, adresse : ${r.address.building} $
{r.address.street}`);
         });
         db.close();
    });
});
```



- Mongoose
 ODM: Object Document Mapping, permet de communiquer avec Mongo avec des objets Entités
- Installation
 npm install mongoose --save
- Schema
 Mongo permet l'absence de schéma, ce qui est peu recommandable dans une utilisation sous la forme d'entité. Mongoose réintroduit ce concept.

NoSQL - MongoDB



Création d'un Schéma

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

var contactSchema = mongoose.Schema({
    firstName: String,
    lastName: String,
})

var Contact = mongoose.model('contact', contactSchema);
```

```
mongoose.connect('mongodb://localhost/addressbook');
var db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function (callback) {
    var contacts = Contact.find(function (err, contacts) {
        if (err) return console.error(err);
        reply({data: contacts});
    });
});
```

NoSQL - Exercice



- · Créer un Model contact avec prenom, nom, email et téléphone
- Utiliser Mongoose pour remplacer le tableau dans l'API REST créé précédemment
- Ajouter des validateurs sur le prenom et nom (champs obligatoire)
- Optionnel : Créer un model Société et lier les Model Société et Contact



Tests Automatisés

Tests automatisés - Introduction



Vérification manuelle

- Ecrire une recette de tests et demander à une personne de la rejouer à des étapes clés (nouvelle version)
- Ecrire le test sous la forme de code, et vérifier visuellement que les résultats attendus soit les bons

Tests automatisés

Le test est codé, la vérification se fait dans un rapport

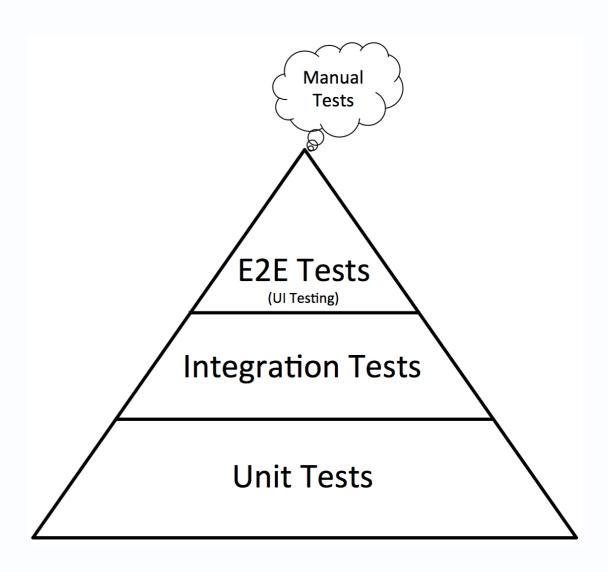
Historique

- sUnit en 1994 (SmallTalk), JUnit en 1997 (Java)
- Les frameworks s'inspirant de jUnit sont catégorisés xUnit (PHPUnit, CUnit...)

Tests automatisés - Pyramide des Tests



- Types de tests
 - Unitaire : tests des méthodes d'une classe
 - Intégration : teste l'intégration entre plusieurs classes
 - Fonctionnels: teste l'application du point de vue du client (HTTP dans le cas du web)
 - End-to-End (E2E): teste l'application dans le client (y compris JavaScript, CSS...)



Tests automatisés - Karma



- ARMA
- Lanceur de test
 Permet de lancer vos tests simultanément dans Chrome, Firefox, Internet Explorer...
- Installation
 npm install -g karma-cli
 npm install karma —save-dev
- Configuration du projet karma init
- Lancement des tests karma start

```
Sourcestagessio (acaaggestio, has alhary adeagostis, sourcesto adaptergossie
Air-de-Romain: Jasmine romain$ karma init
Which testing framework do you want to use ?
Press tab to list possible options. Enter to move to the next question.
> iasmine
Do you want to use Require.js ?
This will add Require.js plugin.
Press tab to list possible options. Enter to move to the next question.
> no
Do you want to capture any browsers automatically ?
Press tab to list possible options. Enter empty string to move to the next question.
> Chrome
> Safari
What is the location of your source and test files ?
You can use glob patterns, eg. "js/*.js" or "test/**/*Spec.js".
Enter empty string to move to the next question.
```

```
Air-de-Romain:Jasmine romain$ karma start

02 09 2015 21:30:11.510:INFO [karma]: Karma v0.13.9 server started at http://localhost:9876/

02 09 2015 21:30:11.518:INFO [launcher]: Starting browser Chrome

02 09 2015 21:30:11.526:INFO [launcher]: Starting browser Safari

02 09 2015 21:30:12.723:INFO [Safari 8.0.7 (Mac OS X 10.10.4)]: Connected on socket HE38slHTBKXL5t5yAAAA with id 54715269

Safari 8.0.7 (Mac OS X 10.10.4): Executed 1 of 1 SUCCESS (0.038 secs / 0.003 secs)

Safari 8.0.7 (Mac OS X 10.10.4): Executed 1 of 1 SUCCESS (0.038 secs / 0.003 secs)

Chrome 45.0.2454 (Mac OS X 10.10.4): Executed 1 of 1 SUCCESS (0.04 secs / 0.008 secs)

TOTAL: 2 SUCCESS
```

Tests automatisés - QUnit



- Créé en 2008 par les développeurs de jQuery
- Type xUnit (JUnit, PHPUnit...): basés sur des assertions
- Plutôt destiné à du code client
- Installation
 npm install --save-dev qunitjs
 bower install --save-dev qunit
- Lancement des tests
 Ouverture du fichier .html grunt-contrib-qunit karma-qunit

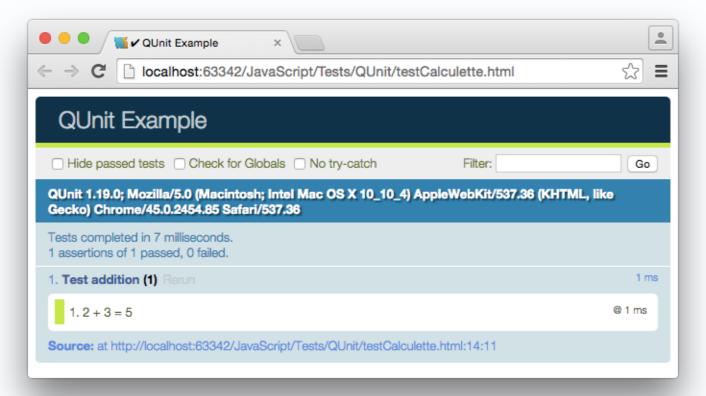


Tests automatisés - QUnit



```
<!-- runner.html -->
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>QUnit Example</title>
    <link rel="stylesheet" href="node modules/qunitjs/qunit/qunit.css">
</head>
<body>
<div id="qunit"></div>
<div id="qunit-fixture"></div>
<script src="calculette.js"></script>
<script src="node_modules/qunitjs/qunit/qunit.js"></script>
<script src="calculette-test.js"></script>
</body>
</html>
```

```
// calculette-test.js
QUnit.test("Test addition", function(assert) {
   assert.equal(calculette.ajouter(2, 3), 5, "2 + 3 = 5");
});
```



Tests automatisés - Jasmine



- Créé en 2010
- Type BDD (Behavior-Driven Development)
- Fonctionne pour le browser ou node.js
- Installation et lancement des tests (node) npm install -g jasmine jasmine init jasmine
- Installation et lancement des tests (browser)
 npm install --save-dev jasmine-core
 SpecRunner.html
 karma



Tests automatisés - Jasmine



```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Jasmine Spec Runner v2.3.4</title>
  <link rel="shortcut icon" type="image/png" href="node modules/jasmine-core/images/</pre>
iasmine favicon.png">
  <link rel="stylesheet" href="node_modules/jasmine-core/lib/jasmine-core/jasmine.css">
  <script src="node modules/jasmine-core/lib/jasmine-core/jasmine.js"></script>
  <script src="node_modules/jasmine-core/lib/jasmine-core/jasmine-html.js"></script>
  <script src="node modules/jasmine-core/lib/jasmine-core/boot.js"></script>
  <!-- include source files here... -->
  <script src="calculette.js"></script>
                                                                                                       <u>.</u>
                                                                    <!-- include spec files here... -->
  <script src="spec/CalculetteSpec.js"></script>
                                                                                                       localhost:63342/JavaScript/Te...
</head>
                                                           3.3.4 3.3.4
                                                                                                Options
<body>
</body>
</html>
                                                          1 spec, 0 failures
                                                                                          finished in 0.002s
                                                            Test calculette
                                                              2 + 3 devraient faire 5
describe("Test calculette", function() {
 it("2 + 3 devraient faire 5", function() {
   expect(calculette.ajouter(2, 3)).toEqual(5);
 });
});
```

Tests automatisés - Mocha



- Créé en 2011
- Type assert ou BDD (le framework est flexible)
- Fonctionne pour le browser ou node.js
- Installation et lancement des tests (node)
 npm install -g mocha
 mocha
- Installation et lancement des tests (browser) npm install -g mocha mocha init npm install chai runner.html karma



Tests automatisés - Mocha



```
<!DOCTYPE html>
<html>
  <head>
    <title>Mocha</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="mocha.css" />
  </head>
  <body>
    <div id="mocha"></div>
    <script src="mocha.js"></script>
    <script src="node modules/chai.js"></script>
    <script>mocha.setup('bdd');</script>
    <script src="src/calculette.js"></script>
                                                                           Mocha Mocha
    <script src="test/calculette-test.js"></script>
    <script>
                                                                           localhost:63342/JavaScript/Te...
      mocha.run();
    </script>
  </body>
                                                                               passes: 1 failures: 0 duration: 0.02s [100%]
</html>
                                                                       Test Addition

√ 2 + 3 devraient faire 5

var assert = chai.assert;
describe('Test Addition', function() {
    it('2 + 3 devraient faire 5', function () {
        assert.equal(5, calculette.ajouter(2, 3));
    });
});
```

Tests automatisés - Voir aussi



- Coverage:
 - Istanbul: https://istanbul.js.org
- Doubles:
 - Sinon: http://sinonjs.org
- Parallélisation des tests :
 - Jest : https://facebook.github.io/jest/
 - AVA: https://github.com/avajs/ava
- Tests End-to-End:
 - Selenium: http://www.seleniumhq.org
 - Webdriver: http://webdriver.io
- PAAS de tests :
 - Sauce Labs: https://saucelabs.com
 - Browser Stack : https://www.browserstack.com