



Université Jendouba

Faculté des Sciences Juridiques
Économiques et de Gestion

Développement Web: JavaScript avancé

AJAX, JQuery et Node.JS

Manel Ben Sassi

Université de Jendouba, FSJEG

bensassi.manel@gmail.com



Sommaire

- 1 Objectifs
- 2 Le client Rich
 - Introduction
 - AJAX
 - Ajax par définition
 - Fonctionnement d'Ajax
 - Objets et méthodes d'Ajax
 - JQUERY
- 3 JavaScript côté serveur : Node.Js
 - Node.js : Définition et architecture
 - Programmation Asynchrone et non bloquante avec Node.js
 - Node.js par la pratique
 - Récupération d'url
 - Paramètres envoyés au niveau de l'url
- 4 Conclusion
- 5 Bibliographie

Les objectifs ...

A la fin de ce cours, vous saurez ...

- Quelques technologies pour le développement des Rich Internet Applications
- Découvrir les aspects avancés du JavaScript côté client mais aussi côté serveur
- La différence entre Ajax et JavaScript
- Les concepts principaux de JavaScript qu'il faut savoir pour utiliser Ajax et Node.js
- Utilités, les avantages et les inconvénients de ces "*nouvelles*" technologies liées au JavaScript



Introduction

Q. Savez-vous la différence entre un site web et une application Web ?

Introduction

Q. Savez-vous la différence entre un site web et une application Web ?

Les applications Web Vs site Web

Site web Une seule fonction : présenter un contenu (Exemple : portail de l'entreprise) ⇒ Interface graphique

Application web Offre des services à travers le site web, consommé à travers le client-navigateur (exemple : chat, mail, entretenir un réseau social, etc)

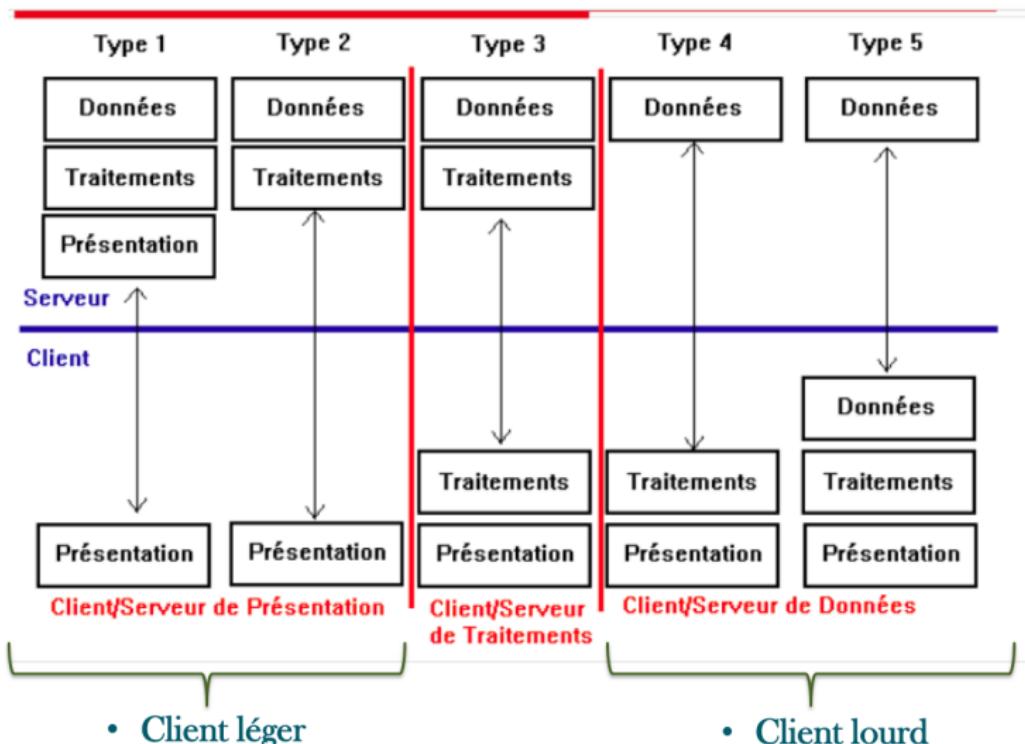


Introduction

Q. Quelles sont les types d'application web qu'on avoir ? et Combien ?

Introduction

Q. Quelles sont les types d'application web qu'on avoir ? et Combien ?



Introduction

Explication des types ...

Type 1 Représente un système Serveur/terminal classique

Type 2 L'affichage effectué par le client se fait à la suite d'un échange de requêtes avec le serveur.

Type 3 Les données restent centralisées mais les traitements sont répartis entre le client et le serveur.

Type 4 et 5 Système popularisé par les SGBDR associés au SQL. Le serveur gère les données, leur intégrité, la sécurité, etc. Le client traite ces données pour éventuellement, en retour, mettre à jour la base.

Introduction

Q. Avez-vous entendu parler des RIA ?

Introduction

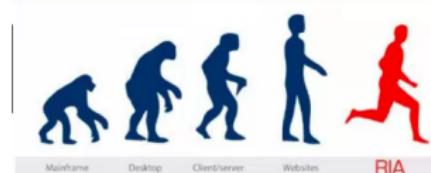
Q. Avez-vous entendu parler des RIA ?

Les applications Internet Riches est

Une application Web qui offre des caractéristiques similaires aux logiciels traditionnels installés sur un ordinateur.

La dimension interactive et la vitesse d'exécution sont particulièrement soignées dans ces applications Web. Elle est exécuté sur un navigateur web (*Aucune installation n'est requise*)

User Interface Evolution



Asynchronous JavaScript and XML



AJAX... des interrogations ?

Q. AJAX est un nouveau langage de programmation ?

AJAX... des interrogations ?

Q. AJAX est un nouveau langage de programmation ?

- **AJAX** n'est ni une technologie ni un langage de programmation
- **AJAX** est un concept de programmation Web reposant sur plusieurs technologies comme le **JavaScript** et le **XML**

AJAX... des interrogations ?

Q. AJAX est un nouveau langage de programmation ?

- **AJAX** n'est ni une technologie ni un langage de programmation
- **AJAX** est un concept de programmation Web reposant sur plusieurs technologies comme le **JavaScript** et le **XML**

Q. Qu'appelle-t-on alors AJAX ?

AJAX est une architecture informatique qui permet de construire des applications web avec une composition de technologie telle que : JS, CSS, DOM, XML, XMLHttpRequest et JSON afin d'améliorer maniabilité et confort d'utilisation des Les applications Riches Internet RIA

AJAX... des interrogations ?

Q. Qui utilise AJAX ?

AJAX... des interrogations ?

Q. Qui utilise AJAX ?

- Les clients Web de messagerie : GMAIL Yahoo Mail, HotMail
- Google Maps
- Bing
- FlickR, Picasa
- Youtube, Dailymotion
- Myspace, Facebook

Pourquoi utiliser AJAX ?

Q. Que permet AJAX de plus ?

Pourquoi utiliser AJAX ?

Q. Que permet AJAX de plus ?

AJAX permet

- Lire les données d'un serveur Web **après le chargement de la page**
- Mettre à jour une partie de la page Web sans recharger la totalité de la page
- Communiquer avec le serveur d'une manière **asynchrone** en échangeant des données **derrière les coulisses**
- Envoyer des données à un serveur Web **en arrière-plan** ⇒ **Diminuer le temps de latence** ⇒ **Augmenter la réactivité de l'application web**



La communication **Synchrone et Asynchrone** ?

Pourquoi utiliser AJAX ?

La communication Synchrone Vs Asynchrone ..

En mode synchrone le processus appelant attend que le processus appelé ait renvoyé sa réponse pour continuer à s'exécuter ⇒
Requête et Réponse

En mode Asynchrone le processus appelant continue à travailler pendant que le processus appelé exécute le traitement demandé et gère via un événement - ou éventuellement via une instruction de synchronisation - le(s) retour(s) du processus appelé.



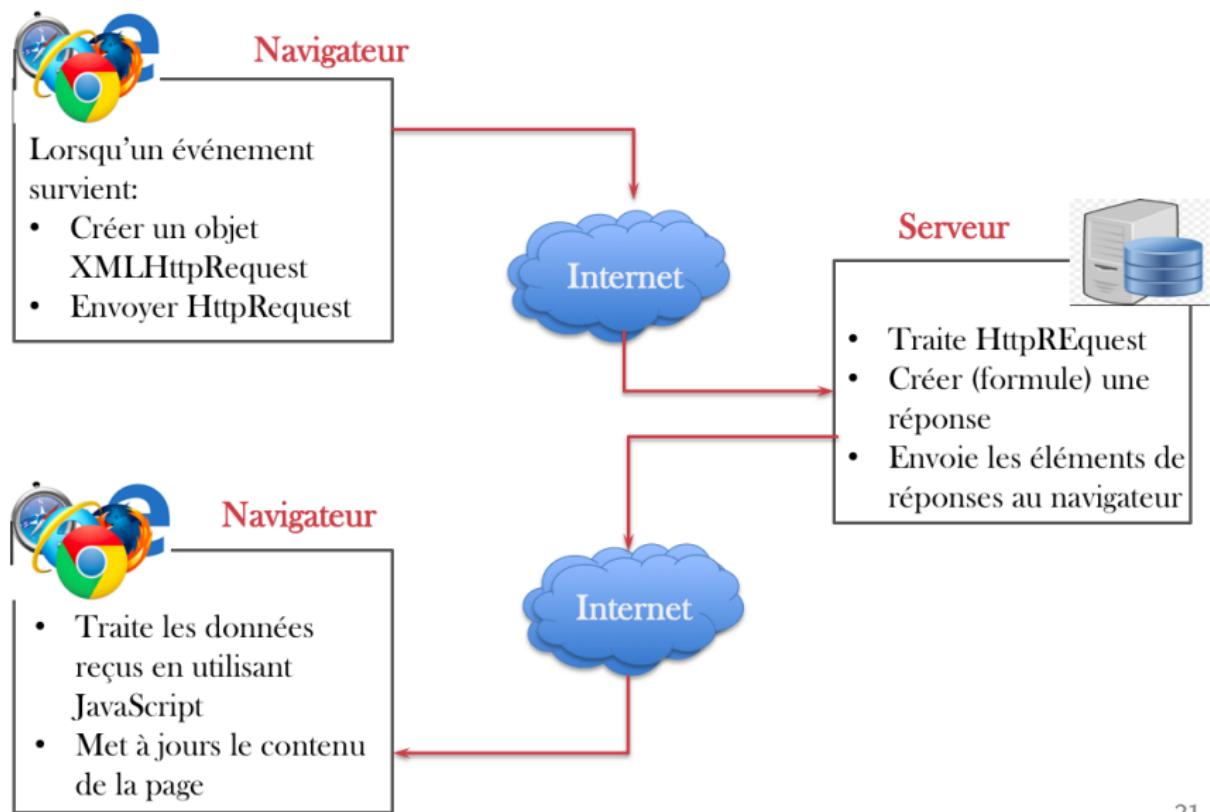
Cool.. C'est quoi au juste Ajax ?

C'est quoi AJAX ?

Asynchronous JavaScript And XML n'est pas un langage de programmation, il est plutôt la combinaison de :

- une requête **XMLHttpRequest**
- **JavaScript** et **HTML DOM** (pour l'affichage et l'utilisation des données)
- *NB* : AJAX peut utiliser le XML lors du transfert des données, comme il peut également les transférer sous la forme Texte ou JSON

Comment fonctionne AJAX ?



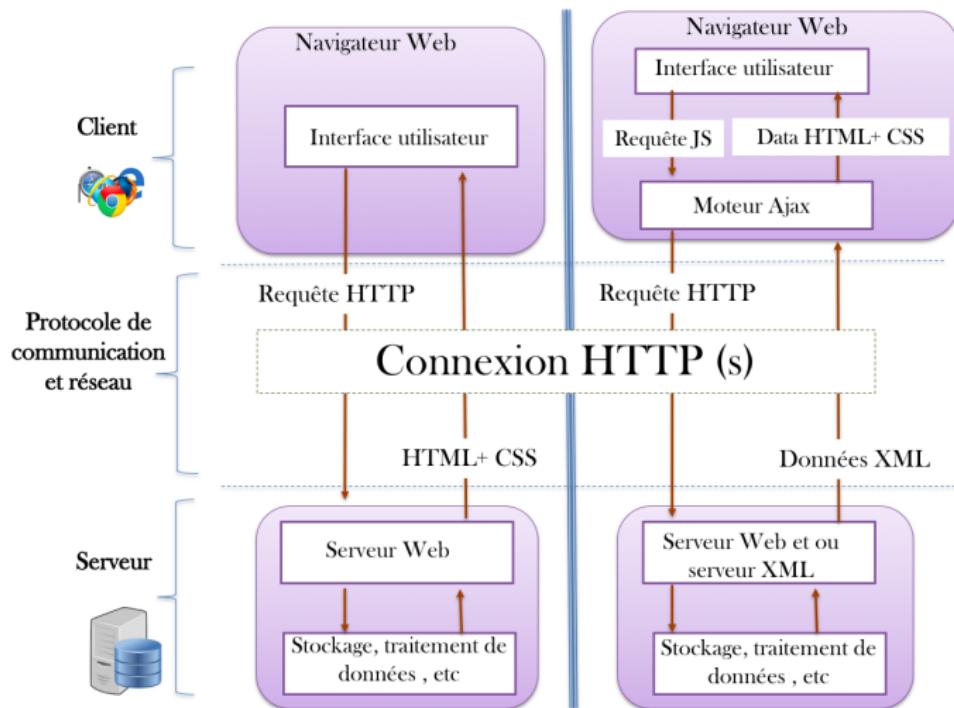
Comment fonctionne AJAX ?

- ① Un évènement survient dans la page web (la page est chargée, un click sur un bouton)
- ② L'objet XMLHttpRequest est créé par le JavaScript
- ③ L'objet XMLHttpRequest envoie une requête au serveur de la page web
- ④ Le serveur traite la requête
- ⑤ Le serveur formule une réponse et l'envoie à la page web
- ⑥ La réponse est lu et est traité par JavaScript côté client
- ⑦ Une action propre est réalisée par JavaScript (comme recharger la page ou rafraîchir l'affichage)

Quelle différence entre le web Classique et Le web avec AJAX ?

Web avec AJAX Vs Web sans AJAX ?

Quelle différence entre le web sans et avec Ajax ?



Les objets et les méthodes d'Ajax



XMLHttpRequest Object

Les méthodes associées à l'objet XMLHttpRequest

Méthodes	Description
<code>XMLHttpRequest()</code>	crée un nouveau objet XMLHttpRequest
<code>abort()</code>	Annule la requête en cours
<code>getAllResponseHeaders()</code>	Retourne toutes les informations de <u>header</u>
<code>getResponseHeader()</code>	Retourne une information spécifiée de <u>header</u>
<code>open(<i>method, url, async, user, pwd</i>)</code>	Spécifie les paramètres de la requête: <ul style="list-style-type: none"> • <u>method</u>: <u>"Get</u> ou <u>"Post</u>" • <u>url</u>: l'<u>url</u> du fichier à traiter • <u>async</u>: true (asynchrone) ou false (synchrone) • User: nom d'utilisateur • Pwd: mot de passe
<code>send()</code>	Envoie la requête au serveur, Utilisé avec la méthode GET
<code>send(<i>string</i>)</code>	Envoie la requête au serveur, Utilisé avec la méthode POST
<code>setRequestHeader()</code>	Ajouter une label ou une valeur à l'entête à envoyer

XMLHttpRequest Object

Les propriétés associées à l'objet XMLHttpRequest

Propriété	Description
onreadystatechange	Définit une fonction à appeler lorsque la propriété readyState change
readyState	Contient le statut de XMLHttpRequest. 0: requête non initialisée 1: connexion établie avec le serveur 2: requête reçue 3: requête en cours de traitement 4: requête finie et réponse prête à envoyer
responseText	Retourne la réponse sous la forme d'une chaîne de caractères
responseXML	Retourne la réponse sous la forme d'un document XML
status	Retourne le statut (nombre) de la requête 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Retourne le statut sous la forme de texte (exemple "OK" ou "Not Found")

XMLHttpRequest Object

Un premier Exemple

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4  <h2>Mon tout premier exemple XMLHttpRequest</h2>
5  <p id="demo">AJAX va changer ce texte!.</p>
6  <button type="button" onclick="loadDoc()">Changer le contenu</button>
7  <script>
8  function loadDoc() {
9      //créer un nouveau objet
10     var xhttp = new XMLHttpRequest();
11     //définir la fonction qui va être exécutée
12     //quand le statut ReadyState change
13     xhttp.onreadystatechange = function() {
14
15         if (this.readyState == 4 && this.status == 200) {
16             //4: requête finie+ réponse prête
17             //status=200 => OK
18             document.getElementById("demo").innerHTML =      this.responseText;
19             //responseText : récupère la réponse sous la forme du texte
20         }
21     };
22     //Spécifier la requête(méthode, url, Asyn)
23     xhttp.open("GET", "ajax_info.txt", true);
24     // envoyer la requête au serveur
25     xhttp.send();
26 }
27 </script>
28 </body>
29 </html>
```

XMLHttpRequest Object

Q. Que choisir Post ou Get comme méthode d'envoi de données ?

XMLHttpRequest Object

Q. Que choisir Post ou Get comme méthode d'envoi de données ?

- Get est plus **rapide** et plus **simple** ⇒ Plus utilisé que Post
- Nous utilisons Post dans les cas suivants :
 - Le fichier caché n'est pas un option (nous devons mettre à jours un fichier ou une BDD sauvegardée au niveau du serveur).
 - Lorsque nous envoyons beaucoup de données au serveur (POST n'a pas de limites de taille).
 - Lorsque nous envoyons les données saisies du formulaire, POST est plus **sécurisé** et plus **robuste** .

Pour conclure, Retenez ...



Ajax pour conclure ...

Les applications AJAX sont

- **des applications 3-tiers (client-serveurs)** : client ↔ AppServer ↔ Source de données.
- **dirigées par les évènements** : clicks des usagers qui peuvent changer les données
- **avec un graphique intensif** effets visuels, un contrôle visuel
- **orientées données** Les usagers manipulent et saisissent les données

JQUERY ...



jQuery .. JS avec plus de simplicité

Ce qu'il faut connaître ... JQuery est une librairie qui permet la manipulation de plusieurs objets :

- HTML-DOM
- CSS
- les évènements et les méthodes associées à HTML
- Effects and animations
- AJAX
-

JQuery .. JS avec plus de simplicité

Ce qu'il faut connaître ... Il y a beaucoup de frameworks JavaScript ... "jQuery" semble être le plus populaire, et aussi le plus extensible. Il est utilisé par les compagnies les plus populaires et puissantes du web telles que :

- Google
- Microsoft
- IBM
- Netflix



jQuery .. JS avec plus de simplicité

jQuery est la bibliothèque qui permet de :

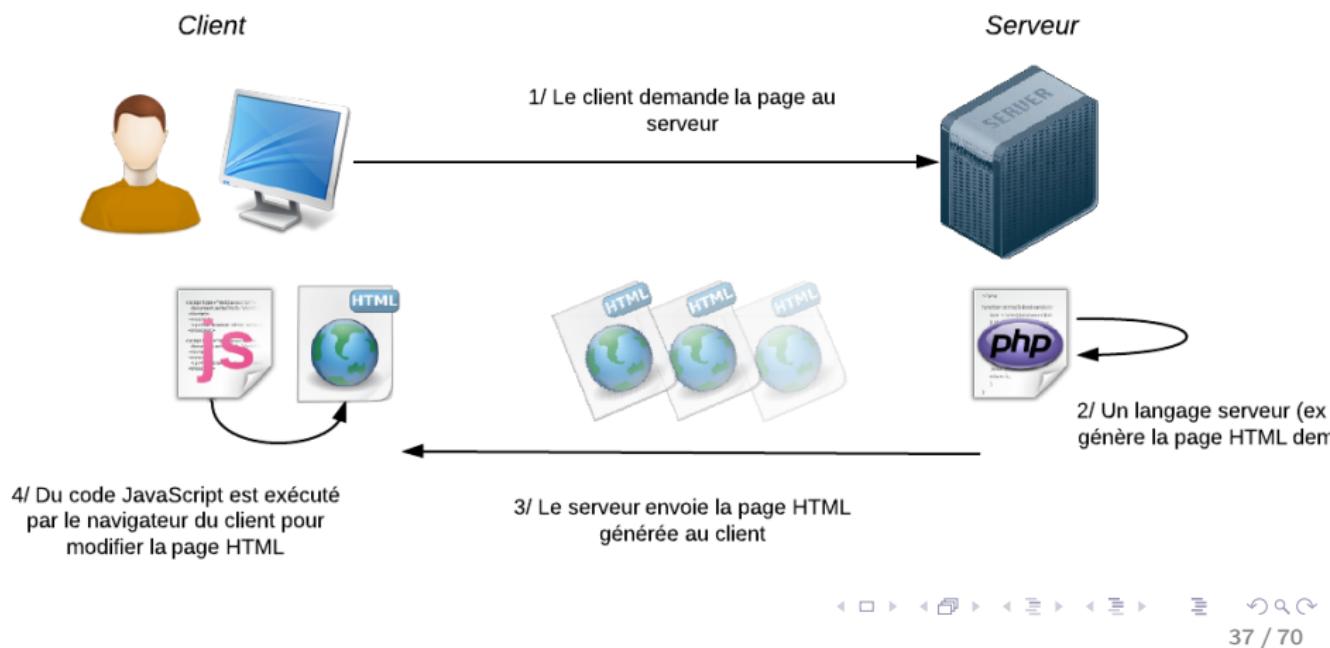
- Ajouter, supprimer ou modifier des éléments HTML au sein d'une page web.
- Changer les styles des éléments de la page en modifiant le CSS qui leur est associé.
- Animer des éléments de la page web.
- Envoyer et recevoir des données depuis un serveur grâce à AJAX (asynchronous JavaScript and XML, c'est-à-dire JavaScript et XML asynchrones) pour ne plus avoir besoin de recharger les pages après validation d'un formulaire.
- Profiter d'une plus grande compatibilité avec les différents navigateurs

Le JavaScript côté Serveur Web ? ...



JS Technologie côté serveur

Le navigateur web du visiteur (Firefox, Chrome, IE...) exécute le code JavaScript et effectue des actions sur la page web.



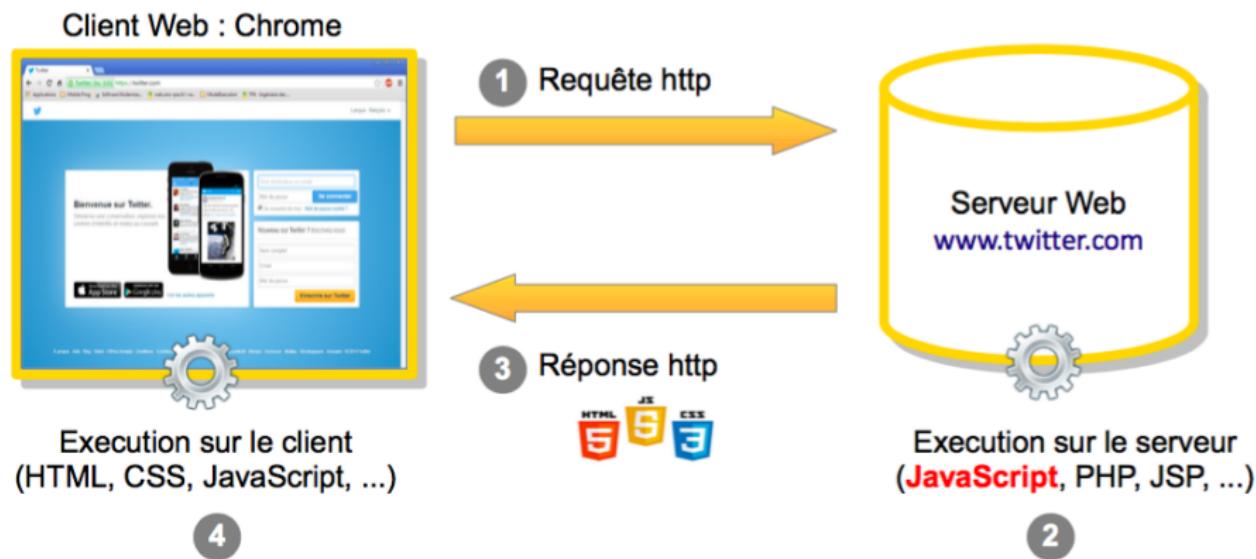
4/ Du code JavaScript est exécuté par le navigateur du client pour modifier la page HTML

3/ Le serveur envoie la page HTML générée au client

2/ Un langage serveur (ex génère la page HTML demandée)

JS Technologie côté serveur

JS Technologie côté serveur



JS Technologie côté serveur : Node.JS

Node.js offre un environnement d'exécution JS côté serveur qui nous permet aussi d'utiliser le langage JavaScript pour générer des pages web et des applications telles que :

- Un serveur de Chat
- Un système d'upload très rapide
- ... et de façon générale n'importe quelle application qui doit répondre à de nombreuses requêtes rapidement et efficacement, en temps réel



Node.js est une plateforme pilotée par les évènements

Node JS : avantages

Grâce au Node.JS

Modèle mono-thread Gestion de la concurrence des clients **differentes** des serveurs multi-threades (saturation) ⇒ Oblige à se tourner vers **un modèle non-bloquant**

Modèle non-bloquant Le fonctionnement asynchrone devient la règle et est tendance ("reactive programming") ⇒ Donne un second souffle au JS

Extensibilité et communauté Il existe de très nombreux paquets (ou modules)

- Gestionnaire de paquets (**npm**)
- Annuaire officiel de paquets :www.npmjs.com

Programmation sur une couche plus basse que d'autres technologies serveur

JS Technologie côté serveur : Node.js

Pourquoi Node.js est rapide ?

JS Technologie côté serveur : Node.js

Pourquoi Node.js est rapide ? Deux raisons ⇒ le moteur V8 et son fonctionnement non bloquant.

Le moteur V8

C'est le moteur d'exécution, **Open Source** ultrarapide V8 de Google Chrome qui analyse et exécute du code JavaScript très rapidement.



La programmation non bloquante du Node.js

La programmation bloquante Vs non bloquante du Node.JS

Dans la programmation bloquante, les actions sont effectuées dans l'ordre :

- ① Le programme commence par télécharger un fichier sur Internet
- ② En deuxième étape, il affiche le fichier à l'utilisateur
- ③ Puis ensuite le programme peut exécuter d'autres instructions (effectuer d'autres actions)

```
request('http://www.site.com/fichier.zip', function (error, response, body) {  
    console.log("Fichier téléchargé!");  
});  
console.log("Je fais d'autres choses en attendant...");
```

La programmation non bloquante du Node.js

La programmation bloquante Vs non bloquante du Node.JS

Dans la programmation non bloquante,

- ① Le programme lance le téléchargement d'un fichier sur Internet
- ② Le programme exécute d'autres instructions (le programme suit son cours)
- ③ Dès que le téléchargement est terminé, le programme retourne vers l'instruction en question pour effectuer les actions demandées ⇒ il affiche le fichier

```
var callback = function (error, response, body) {
    console.log("Fichier téléchargé !");
};

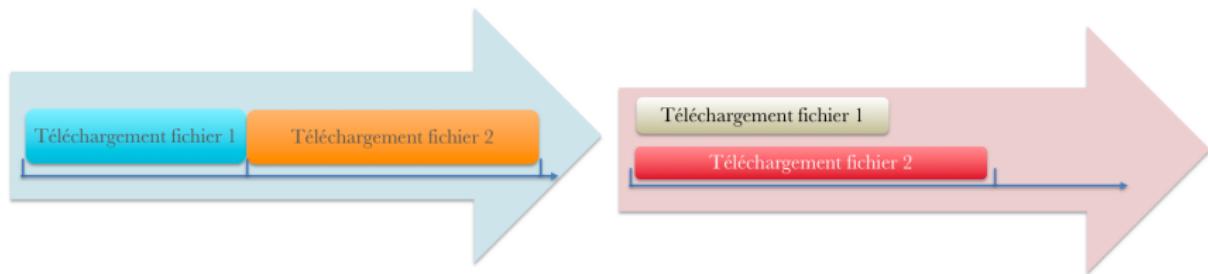
request('http://www.site.com/fichier.zip', callback);
request('http://www.site.com/autrefichier.zip', callback);
```

JS Technologie côté serveur : Node.js

La programmation Asynchrone et non bloquante du Node.JS

Par définition,

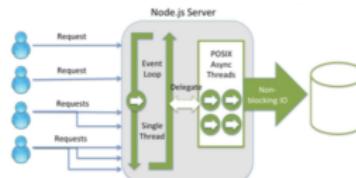
- La plupart des opérations I/O en JavaScript sont non bloquantes : requêtes HTTP, requêtes AJAX, lecture-écriture sur le système de fichier, etc.
- ⇒ Les opérations s'exécutent de manière linéaire sans attendre la fin de l'opération précédente. Les fonction de **callback** permettent de récupérer la réponse d'une opération.



JS Technologie côté serveur : Node.js

D'une manière plus concrète, dans un environnement non bloquant,

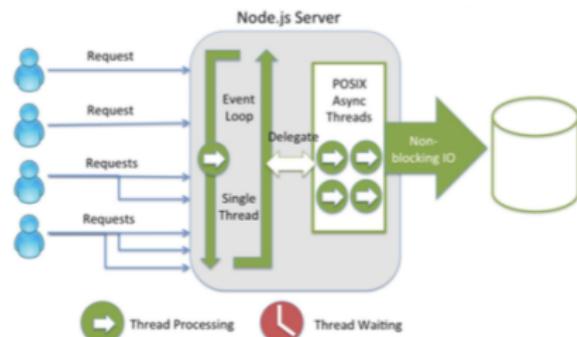
- ▷ Node.js dispose en son sein d'une boucle d'évènements (event loop) qui va continuellement traiter les évènements reçus.
- ▷ Dans le cas d'un serveur web avec Node.js, le serveur va conserver une pile de fonctions à exécuter (callback) lors de la réception d'évènements précis.
- ▷ L'unique thread de Node.js va traiter les évènements apparaissant dans la queue d'évènements les uns après les autres, au fil des requêtes et fin d'IO.
- ▷ Permettre de traiter des milliers de requêtes simultanées en consommant très peu de mémoire.



JS Technologie côté serveur : Node.js

Node.js Synthèse des particularités

- Piloté par les évènements.
- Modèle d'entrées / sorties (I/O) asynchrone non bloquant en passant par l'utilisation de callbacks.
- Tout à l'intérieur de Node.js est traité par un seul et unique thread.
- Approche réactive implémentée par le boucle d'évènements



Node.js par la pratique

CODER AND TESTER



Node.js par la pratique



Node.js par la pratique

Installation de Node.js

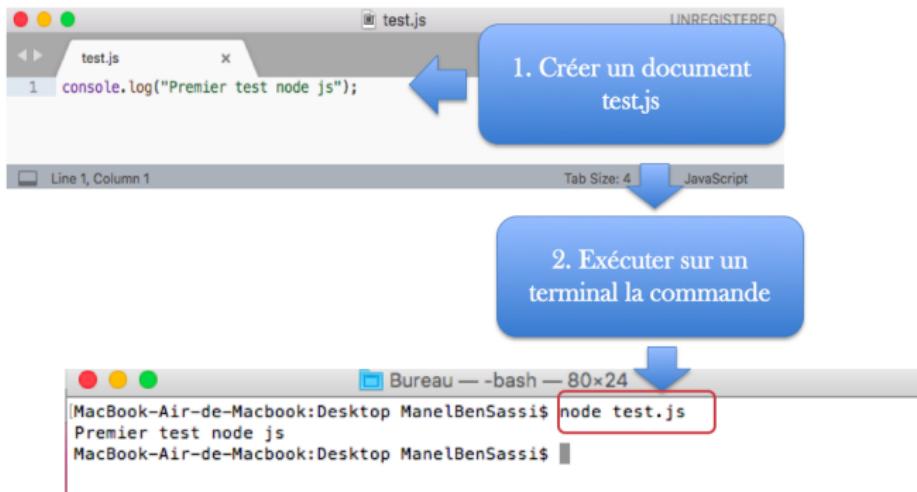
- ① Visitez le lien : <https://nodejs.org/en/>
- ② Télécharger la version stable
- ③ Exécuter la commande `node -v` dans votre terminal pour vérifier que l'installation s'est bien déroulée..

The screenshot shows the official Node.js website. At the top is a dark header with the Node.js logo and navigation links: HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, NEWS, and FOUNDATION. Below the header is a dark section with white text: "Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world." A green callout box contains the text "Important upcoming security releases, please review". Below this is a "Download for macOS (x64)" section. It features two prominent buttons: a green one for "8.9.2 LTS" (labeled "Recommended For Most Users") and a blue one for "9.2.0 Current" (labeled "Latest Features"). Smaller links below the main download buttons include "Other Downloads", "Changelog", "API Docs", "Other Downloads", "Changelog", and "API Docs". A note says "Or have a look at the [LTS schedule](#)." At the bottom, there's a link to "Sign up for Node.js Everywhere, the official Node.js Weekly Newsletter."

A screenshot of a Mac OS X terminal window titled "macbookair — bash — 80x17". The window shows the command "node -v" being run and the output "v6.9.1". The terminal interface includes standard Mac OS X window controls and a scroll bar.

Node.js par la pratique

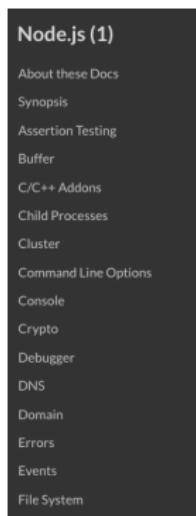
Node.js un petit test



Node.js par la pratique

Les modules de Node.js

- Node.js est organisé en plusieurs modules.
- Chaque module gère une fonctionnalité du "core" : système de fichier, réseau (DNS, HTTP, TCP, UDP et TLS), fonctions cryptographiques, etc.
- Les modules sont chargés dans le code avec l'instruction `require()`.



Node.js v5.12.0 Documentation

[Index](#) | [View on single page](#) | [View as JSON](#)

Table of Contents

- [About these Docs](#)
- [Synopsis](#)
- [Assertion Testing](#)
- [Buffer](#)
- [C/C++ Addons](#)
- [Child Processes](#)
- [Cluster](#)
- [Command Line Options](#)
- [Console](#)
- [Crypto](#)
- [Debugger](#)
- [DNS](#)
- [Domain](#)

Node.js par la pratique

Serveur Web Apache Vs Serveur Node.js

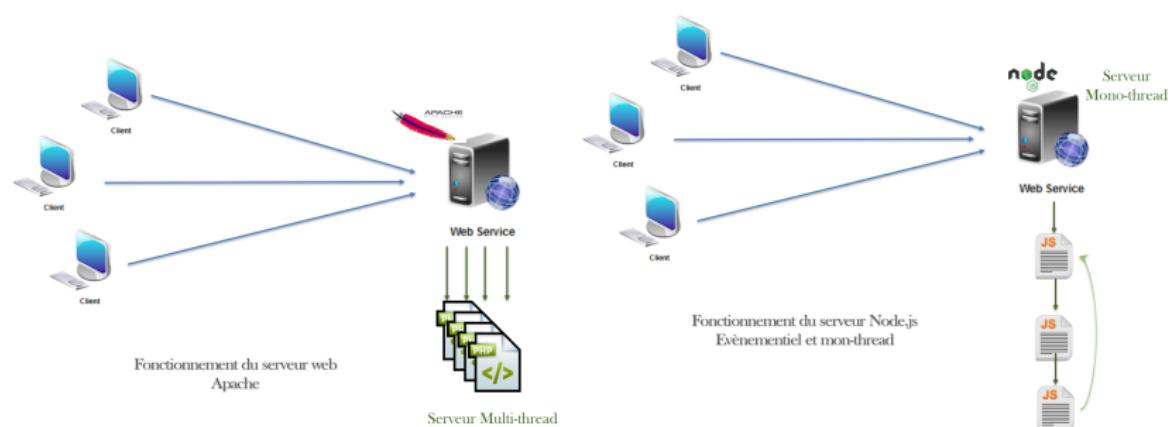


Figure – Nous pouvons toujours construire un serveur web avec Node.js

Node.js par la pratique

```
var http = require('http');
var server = http.createServer(function(req, res) {
    res.writeHead(200);
    res.end('Salut tout le monde !');
});
server.listen(8080);
```

createServer: création d'un serveur:
d'une entité qui reste à l'écoute d'une
requête (l'objet req) et formule une réponse
(l'objet res) à travers une fonction

- La fonction **createServer** prend en paramètre une fonction **callback** avec 2 paramètres : **req** sauvegarde tout ce que le visiteur souhaite savoir et **res** contient la réponse à afficher.
- On envoie le code "200" dans l'en-tête de la réponse, qui signifie au navigateur "OK" selon le protocole HTTP.
- La méthode **end()** marque la fin de la réponse.
- On lance un processus qui reste à l'écoute sur le port 8080

Node.js par la pratique

Si on veut afficher une image ou une page HTML ?

Node.js par la pratique

Si on veut afficher une image ou une page HTML ?

Il faut connaître que le protocole HTTP exige que :

- le serveur doit indiquer le type de données, appelé **MIME** qu'il s'apprête à envoyer au client soit :
 - ▷ 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**
- Ces informations sont envoyées dans l'en-tête de la réponse du serveur.



Exemple ?

Node.js par la pratique

Un premier exemple

The screenshot illustrates the execution flow from a code editor to a browser. A red arrow points from the code editor window down to the browser window.

Code Editor (serverNodeJS 2.js):

```
1 var http = require('http');
2
3 var server = http.createServer(function(req, res) {
4     res.writeHead(200, {"Content-Type": "text/html"});
5     res.end('<p> Mon premier exemple en <strong>HTML</strong> !</p>');
6 });
7 server.listen(8080);
```

Line 6, Column 4 is highlighted.

JavaScript is selected as the language.

Browser (localhost:8080):

The browser shows the rendered HTML output: **Mon premier exemple en **HTML** !**

Node.js par la pratique

Un deuxième exemple

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

var server = http.createServer(function(req, res) {
    res.writeHead(200, {"Content-Type": "text/html"});
    res.write('<!DOCTYPE html>' +
    '<html>' +
    '<head>' +
    '<meta charset="utf-8" />' +
    '<title>Ma page Node.js !</title>' +
    '</head>' +
    '<body>' +
    '<p>Ceci mon premier test en page <strong>HTML</strong> !</p>' +
    '</body>' +
    '</html>');
    res.end();
});
server.listen(8080);
```



Ceci mon premier test en page HTML !

res.write : permet d'écrire la réponse en plusieurs temps !

Node.js par la pratique

Questions

- *Comment peut-on récupérer la page demandée ? (exemple une page sous le dossier/dossier/mapage, /page.html, etc)*
- *Comment peut-on récupérer les paramètres qui circulent au niveau de l'URL ? (exemple http://localhost:8080/test?nom=Tounsi&prenom=Med)*



Node.js par la pratique

La page demandée par l'usager



- Recharger un nouveau module `url` : `var url = require("url");`
- Parser la requête de l'utilisateur `pathname` :
`url.parse(req.url).pathname;`

Node.js par la pratique

La page demandée par l'usager : le code source

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

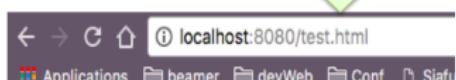
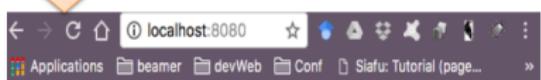
var server = http.createServer(function(req, res) {
    var page = url.parse(req.url).pathname;
    console.log(page);
    res.writeHead(200, {"Content-Type": "text/plain"});
    if (page == '/') {
        res.write('Bonjour, Vous êtes à l\'accueil, que puis-je pour vous ?');
    }
    else if (page == '/test.html') {
        res.write('Bonjour! Vous êtes dans la première page test!');
    }
    else if (page == '/web/test1.html') {
        res.write('HeHo Vous êtes dans la deuxième page test!!!');
    }
    res.end();
});
server.listen(8080);
```

Node.js par la pratique

La page demandée par l'usager : le code source et l'exécution

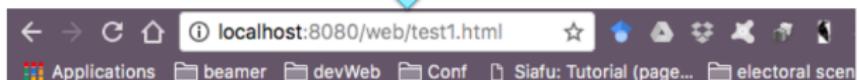
```
var http = require('http');
var url = require('url');

var server = http.createServer(function(req, res) {
    var page = url.parse(req.url).pathname;
    console.log(page);
    res.writeHead(200, {"Content-Type": "text/plain"});
    if (page == '/') {
        res.write('Bonjour, Vous êtes à l\'accueil, que puis-je pour vous ?');
    }
    else if (page == '/test.html') {
        res.write('Bonjour! Vous êtes dans la première page test!');
    }
    else if (page == '/web/test1.html') {
        res.write('HeHo Vous êtes dans la deuxième page test!!');
    }
    res.end();
});
server.listen(8080);
```



Vous êtes à l'accueil, que puis-je pour vous ?

Vous êtes dans la première page test!



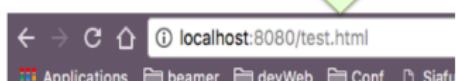
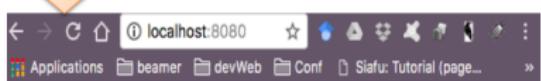
HeHo Vous êtes dans la deuxième page test!!

Node.js par la pratique

La page demandée par l'usager : le code source et l'exécution

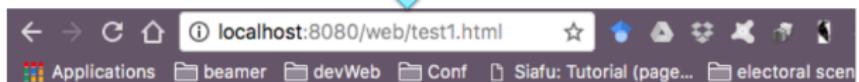
```
var http = require('http');
var url = require('url');

var server = http.createServer(function(req, res) {
  var page = url.parse(req.url).pathname;
  console.log(page);
  res.writeHead(200, {"Content-Type": "text/plain"});
  if (page == '/') {
    res.write('Bonjour, Vous êtes à l\'accueil, que puis-je pour vous ?');
  }
  else if (page == '/test.html') {
    res.write('Bonjour! Vous êtes dans la première page test!');
  }
  else if (page == '/web/test1.html') {
    res.write('HeHo Vous êtes dans la deuxième page test!!');
  }
  res.end();
});
server.listen(8080);
```



Vous êtes à l'accueil, que puis-je pour vous ?

Vous êtes dans la première page test!



HeHo Vous êtes dans la deuxième page test!!

Node.js par la pratique

- Récupérer toute la chaîne contenant les variables `query` :
`url.parse(req.url).query`
- Recharger un module qui permet d'interroger la chaîne récupérée `querystring` : `var querystring = require('querystring');`
- Récupérer les chaînes dans un tableau : `var params = querystring.parse(url.parse(req.url).query);`
- Pour récupérer les éléments du tableau, il faut écrire par exemple `params['prenom']`

```
querystring.parse(url.parse(req.url).query)['param2']  
url.parse(req.url).pathname  

```

http://localhost:8080/chemin/vers/la/page?param1=valeur¶m2=valeur

url.parse(req.url).query

Node.js par la pratique

Les paramètres envoyés : le code source

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

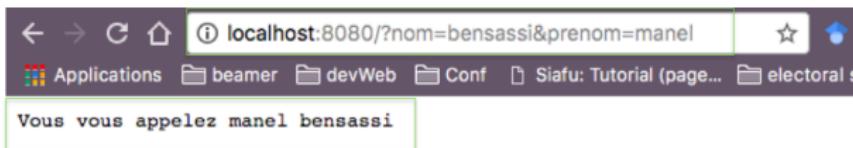
var server = http.createServer(function(req, res) {
  var params = querystring.parse(url.parse(req.url).query);
  res.writeHead(200, {"Content-Type": "text/plain"});
  if ('prenom' in params && 'nom' in params) {
    res.write('Vous vous appelez ' + params['prenom'] + ' ' + params['nom']);
  }
  else {
    res.write('Vous devez bien avoir un prénom et un nom, non ?');
  }
  res.end();
});
server.listen(8080);
```

Node.js par la pratique

Les paramètres envoyés : le code source et l'exécution

```
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('Vous devez bien avoir un prénom et un nom, non ?');
  }
  res.end();
});
server.listen(8080);
```



Pour conclure, l'étape suivante ...



L'étape suivante ...

Maintenant, il est temps de découvrir les aspects les plus avancés de Node.J

- Le module NPM pour ajouter vos propres modules JavaScript
- Le framework Express.js
- Interaction de Node.js avec la base de données non Structurée NoSQL et MongoDB



Références Bibliographiques



Références Bibliographiques

- ① Bien développer pour le Web 2.0 : Bonnes pratiques Ajax, Christophe Porteneuve, 2^{me} Edition Eyrolles, 674 pages
- ② JavaScript et Ajax pour les nuls, Andy Harris, Edition First, 420 pages
- ③ Sites web interactifs : JavaScript, Ajax, jQuery, Edition Micro Application, 820 pages
- ④ Programmation avec Node.js, Express.js et MongoDB : JavaScript côté serveur, Eric Sarrion , Edition Eyrolles, 586 pages, 2014