

# Node et technologies HTML5

Florent Marchand de Kerchove  
Merwan Achibet

UFR sciences et techniques  
Université du Havre

9 novembre 2011

# Introduction



## 1 Node et compagnie

Node

Express

npm

## 2 HTML5

WebSocket et Socket.IO

Canvas HTML

## 3 Exercice

Présentation

Couleurs

Tracé

À vous

# Node

## Description

<http://nodejs.org/>

Serveur :

- événementiel,
- asynchrone,
- en JavaScript.

Intérêts :

- Language identique client/serveur
- Performance élevée (100 000+ connexions simultanées)
- API réseau élémentaire
- Riche librairie de modules

Similaire à Twisted ou EventMachine

# Node

## Serveur événementiel

Reçoit des événements et y répond

- Semblable au DOM et à jQuery
- Fortement adapté aux applications réseau
- Approche dynamique

```
server.on('join', function(user) { join(user); });  
server.on('message', function(msg) { broadcast(msg); });  
server.on('leave', function(user) { leave(user); });
```

# Node

## Serveur asynchrone

Approche non-bloquante :

- Entrées/sorties asynchrones (*epoll*, *kqueue*, ...)
- Le processus dort en attendant les événements
- Aucun *thread*, un seul processus (*fork* possible)
- Plus simple à programmer

Inconvénient :

- Éviter les appels bloquants trop longs

# Node

## Serveur en JavaScript

Utilise le moteur JavaScript V8 de Google (Ecma-262 édition 5)

Avantages du JavaScript côté serveur :

- Adapté au modèle événementielle
- Fonctionnalités asynchrones incluses
- Simplifie les échanges client-serveur
- Partage de code possible

# Node

## Installation manuelle

Prérequis :

```
sudo apt-get install build-essential git libssl-dev
```

Dépôt officiel : <http://github.com/joyent/node/>

```
cd  
git clone http://github.com/joyent/node.git  
cd node  
git checkout v0.4.12  
./configure  
make  
sudo make install
```



# Node

## Exemple 1

### Serveur écho

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

var server = net.createServer(function (socket) {
  socket.write("Echo server");
  socket.pipe(socket);
}).listen(1337);
```

# Node

## Exemple 2

### Serveur HTTP

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

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World');
}).listen(1337);
```

```
ab -n 1000 -c 1000 http://192.168.1.6:1337/
```

# Node

## Exemple 3

### Serveur HTTP bloquant

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

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  setTimeout(function(){
    res.end('Hello World');
  }, 2000)
}).listen(1337);
```

```
ab -n 1000 -c 1000 http://192.168.1.6:1337/
```

# Node

## Exemple 4

### Serveur chat TCP

```
var net = require('net');
var sockets = [];

net.Server(function(socket) {
  sockets.push(socket);
  socket.on('data', function(data) {
    sockets.forEach(function(s) {
      s.write(socket.fd + '> ' + data);
    });
  });
  socket.on('end', function() {
    sockets.splice(sockets.indexOf(socket), 1);
  });
}).listen(1337);
```

# Express

## Description

[http ://expressjs.com/](http://expressjs.com/)

Fonctionnalités essentielles pour des serveurs web :

- Routes et verbes HTTP
- Authentification
- Gestion de session
- Support des templates HTML (Haml, Jade, ...)
- Cache automatique
- etc.

# Express

## Exemple 1

### Création d'un serveur

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

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

app.listen(1337);
```

```
curl http://192.168.1.6:1337/
```

# Express

## Exemple 2

Chemins basés sur des *regexps*

### Routes

```
app.get('/potion/:name', function(req, res){  
  res.send('potion ' + req.params.name);  
});  
  
app.get('/spell/:school/:name', function(req, res){  
  res.send(req.params.name + ' - ' +  
    req.params.school + ' magic');
```

```
curl http://192.168.1.6:1337/potion/love  
curl http://192.168.1.6:1337/spell/fire/meteor
```

# npm

## Description

<http://npmjs.org/>

node package manager

- Installe et met à jour des modules pour node
- Recherche de modules par description, mots-clés
- Gère les dépendances automatiquement
- Aide au développement de modules

## Installation

```
curl http://npmjs.org/install.sh | sh
```



# npm

## Utilisation

### Installer un module

```
npm install express [-g]
```

### Installation locale par défaut :

```
$ tree -dL 2 node_modules/  
node_modules/  
├─ express  
│  ├── bin  
│  ├── lib  
│  ├── node_modules  
│  └─ testing
```

- Versions différentes pour chaque projet
- Binaires disséminés

À chaque paquet sa description en JSON :

package.json

```
{
  "name": "magic",
  "version": "1.2.3",
  "description": "Enhance the magic possibilities of node",
  "author": "Merlin <merlin@camelot.co.uk>",
  "dependencies": {
    "knights-of-the-round": "2.x",
    "joust": ">= 1.8.1"
  }
}
```

Installer et mettre à jour toutes les dépendances du projet courant

```
npm install  
npm update
```

Charger le projet courant sur le NODE\_PATH

```
npm link
```

À vous la gloire !

```
npm adduser  
npm publish
```

# Node et compagnie

## Ressources et documentation

### Node :

- <http://nodejs.org/>
- <http://github.com/joyent/node/wiki/>
- <http://nodejs.org/docs/v0.4.12/api/>
- <http://howtonode.org/>

### Express :

- <http://expressjs.com/>
- <http://expressjs.com/guide.html>
- <http://github.com/visionmedia/express/wiki>

### npm :

- <http://npmjs.org/>
- `'man npm'`

## 1 Node et compagnie

Node

Express

npm

## 2 HTML5

WebSocket et Socket.IO

Canvas HTML

## 3 Exercice

Présentation

Couleurs

Tracé

À vous

# Protocole WebSocket

## Description

Communication bidirectionnelle persistante :

- Réponse du standard aux techniques Comet
- Rend obsolètes HTTP *long-polling* et HTTP *streaming*
- Véritable *full-duplex* entre client et serveur HTTP
- Utilise la connexion TCP créée pour une requête HTTP

Communication efficace :

- Idéal pour recevoir des notifications du serveur
- Meilleure solution pour des applications temps-réel (bourse, jeux, ...)

# Socket.IO

## Description

Module pour node qui implémente différentes techniques pour une connexion *full-duplex* :

- WebSocket
- Flash socket
- HTTP *long-polling* et *streaming*
- ...

Choisit la meilleure méthode supportée par le navigateur

Ajoute des fonctionnalités utiles :

- *Heartbeats, timeouts*
- *Namespaces*
- *Acknowledgments*

# Socket.IO

## Utilisation

Partie serveur avec Express :

```
var app = require('express').createServer();
var io = require('socket.io').listen(app);

io.sockets.on('connection', function(socket) {
  socket.on('message', function(msg) {
    socket.broadcast.emit('message', {
      from: socket.id,
      msg: msg
    });
  });
});
```



# Socket.IO

## Utilisation

Partie client :

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect();
  socket.send('hi');
  socket.on('message', function(data) {
    console.log(data.from + '> ' + data.msg);
  });
</script>
```

# Canvas HTML

## Description

Élément `<canvas>` :

- Dessin et animation sur une page web
- Contextes 2d et 3d (WebGL)
- Implémenté dans les navigateurs majeurs

Alternative aux SVG plus performante :

- Surface *bitmapped* plutôt que vectorielle
- Pas d'insertion dans le DOM
- Accélération matérielle possible

# Canvas HTML

## Utilisation

### HTML

```
<canvas id="canvas" width="300" height="300">  
  Canvas non supporté par le navigateur  
</canvas>
```

### JavaScript

```
var canvas = document.getElementById('canvas');  
var ctx = canvas.getContext('2d');
```

# Canvas HTML

## Exemple 1

### Remplissage de zone

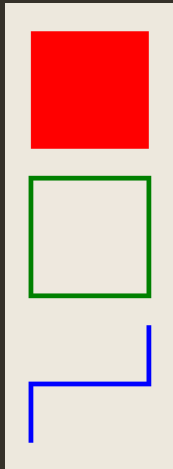
```
ctx.fillStyle = 'red';  
ctx.fillRect(25, 25, 100, 100);
```

### Contour de zone

```
ctx.strokeStyle = 'green';  
ctx.strokeRect(25, 150, 100, 100);
```

### Tracé

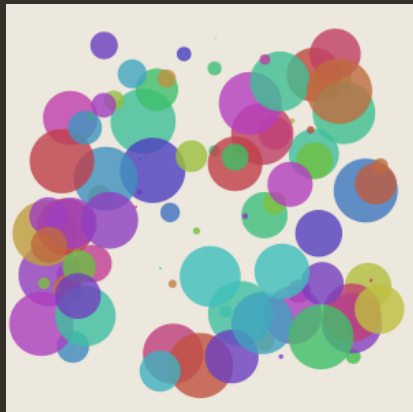
```
ctx.strokeStyle = 'blue';  
ctx.beginPath();  
ctx.moveTo(125, 275);  
ctx.lineTo(125, 325);  
ctx.lineTo(25, 325);  
ctx.lineTo(25, 375);  
ctx.stroke();
```



# Canvas HTML

## Exemple 2

```
ctx.fillStyle = 'hsl(40, 30%, 90%)';  
ctx.fillRect(0, 0, 300, 300);  
  
for (var i=0; i < 100; ++i) {  
    ctx.save();  
    ctx.translate(Math.random()*250,  
                  Math.random()*250);  
    ctx.fillStyle = 'hsla(' +  
        Math.random()*360 +  
        ', 50%, 50%, 0.8)';  
    ctx.beginPath();  
    ctx.arc(24, 24,  
            12 + 12*Math.sin(i),  
            Math.PI*2, false);  
    ctx.fill();  
    ctx.restore();  
}
```



## 1 Node et compagnie

Node

Express

npm

## 2 HTML5

WebSocket et Socket.IO

Canvas HTML

## 3 Exercice

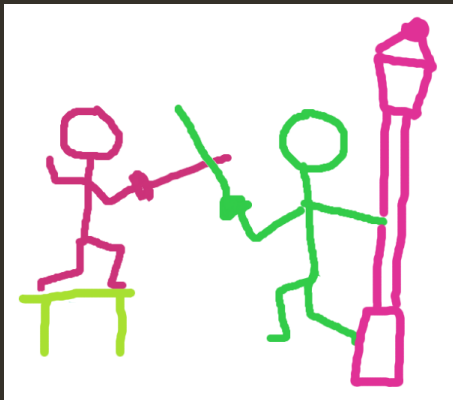
Présentation

Couleurs

Tracé

À vous

# Démonstration



► <http://192.168.1.6:8080/>

# Couleurs

## Modèle HSL



HSL :

- Hue ( $0 \leq H < 360$ )
- Saturation (%)
- Luminance (%)

Dans notre application :

- Chaque client a une teinte différente
- Saturation et luminance fixée



# Couleurs

## Synchronisation

Données serveur :

```
hues = {  
    1234567: 120,  
    7687554: 265,  
    9085231: 27  
}
```

Données client :

```
myHue = 120  
otherHues = {  
    7687554: 265,  
    9085231: 27  
}
```

Connexion :

- Attribution d'une teinte et broadcast

Connexion :

- Réception et enregistrement de la teinte

Déconnexion :

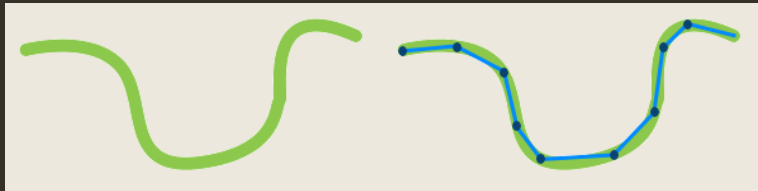
- Suppression de la teinte et broadcast

Déconnexion :

- Suppression des données liées aux clients

# Tracé

## Segment par segment



Comment communiquer un tracé ?

- Un tracé = plusieurs segments
- Une propriété du canvas (`lineCap`) permet d'adoucir les traits
- On transmet en continu le dernier segment tracé par la souris

# Tracé

## Synchronisation

Un client local :

- Surveille les événements `mousedown` et `mouseup`
- Enregistre la position  $p$  du curseur à chaque `mousemove`
- $(p_{t-1}, p_t) = \text{segment}$
- Dessine le segment sur le canvas
- Transmet le segment au serveur

Le serveur :

- Broadcast le segment à tous les autres clients

Un client distant :

- Reçoit le segment et l'identifiant du dessinateur
- Détermine la teinte du segment
- Dessine le segment sur son canvas

# Démarrage

Pour récupérer le squelette du projet :

```
git clone -b base http://github.com/merwaaan/multicanvas.git  
cd multicanvas  
npm install
```

A vous de jouer !

# Ressources et documentation

## Canvas :

- <http://developer.mozilla.org/en/HTML/Canvas>
- [http://developer.mozilla.org/en/Canvas\\_tutorial](http://developer.mozilla.org/en/Canvas_tutorial)
- <http://dev.opera.com/articles/view/html-5-canvas-the-basics/>

## Socket.IO

- <http://socket.io>
- <http://github.com/learnboost/socket.io/wiki/>