Node.js chat with Socket.io

Initialiser l'espace de travail

Commencez par créer un nouveau dossier et initialiser votre espace de travail en créant le fichier `package.json`, grâce à l'utilitaire `npm init`.

Installer express

Dans un premier temps, nous allons mettre en place le serveur express statique, qui s'occupera de l'envoi des fichiers CSS, JS, images, … etc aux clients.

Via npm, installez express (sans oublier l'option `--save`).

Création du serveur statique

Une fois express installé, créez le fichier principal `index.js` à la racine du répertoire, et écrivez le code nécessaire au démarrage d'un serveur express.

Vous écouterez les connexions sur le port 9000.

Votre serveur doit implémenter un middleware permettant de renvoyer les ressources statiques du dossier `/public/` (que vous créerez également à la racine).

Les ressources à votre disposition pour cela sont :

- Exemple Hello world
- Servir des fichiers statiques avec Express

Afin de vérifier que votre serveur statique se comporte correctement, créez un fichier `/public/scripts/app.js` contenant une simple instruction :

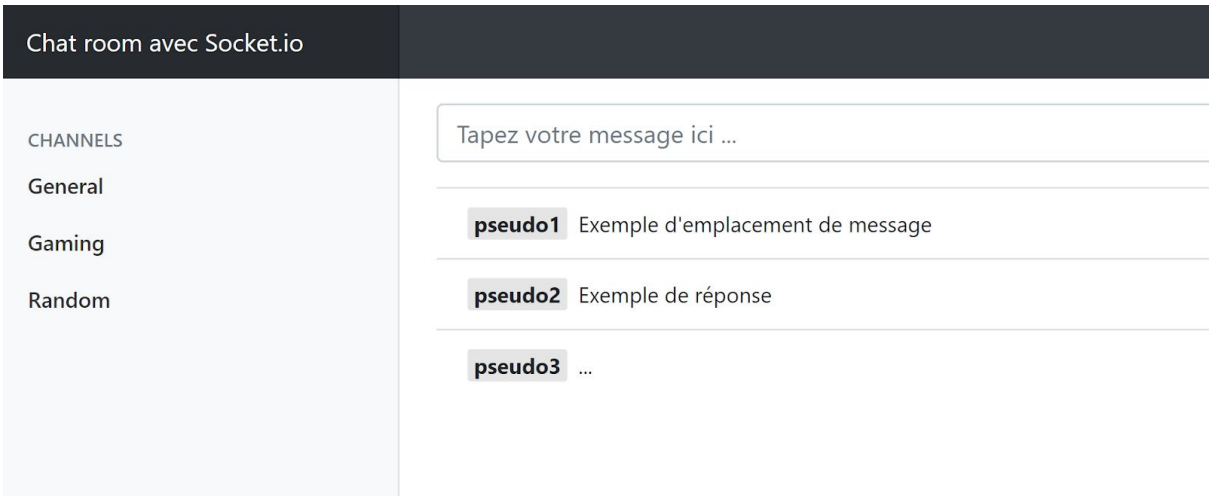
```
'use strict';
console.log("Hello World!");
```

… et vérifiez que http://localhost:9000/scripts/app.js vous retourne bel et bien le contenu de ce fichier.

Création de l'interface de chat

Nous allons créer une page d'accueil contenant l'interface du chat, afin de permettre à nos clients de pouvoir discuter entre eux.

Cette page comportera une zone principale avec un champs texte pour envoyer des messages, et la discussion générale en dessous.
Une zone secondaire à gauche représentera la liste des channels disponibles :



Installation de pug

En vous aidant de la documentation officielle, installez le moteur de templating Pug dans votre projet, et créez le fichier `/views/index.pug` dans votre projet.

Paramètrez ensuite votre application Express afin de lui dire que le moteur de templating utilisé sera Pug.

Enfin, créez la route Express qui écoute sur `/` et qui rend simplement votre fichier `index.pug`

Lorsque Pug sera configuré, vous prendrez le temps de fabriquer une page HTML et CSS élégante et moderne pour votre chat.
Vous pouvez prendre exemple sur le modèle de l'image plus haut, et l'adapter au grés de vos envies (en conservant toutefois les zones décrites).

Installer Socket.io

Socket.io est une librairie permettant de manipuler les sockets en JavaScript.
Elle offre une API pour le serveur Node.js **et aussi** pour le client (dans le navigateur).

Exemple de mise en place

Côté serveur

La mise en place d'un serveur websocket sur un serveur Express est très simple avec Socket.io. Cet exemple de code montre comment nous pouvons procéder :

```
const app = express() // Créer l'application Express
const server = http.Server(app) // Récupère l'objet Server de l'app Express
const io = require('socket.io')(server) // Créer une instance de socket.io
sur ce serveur

...

io.on('connection', (socket) => {
   console.log('Client', socket.id, 'is connected via WebSockets')
})

...

// Ecoute sur l'objet Server (incluant l'app Express et Socket.io)
server.listen(PORT, () => console.log(`✓ Le serveur écoute sur le port
${PORT}`))
```

Côté client

Lorsque Socket.io est mit en place sur un serveur HTTP, une route statique vers `/socket.io/socket.io.js` est automatiquement créée pour servir la bibliothèque **client** de Socket.io.

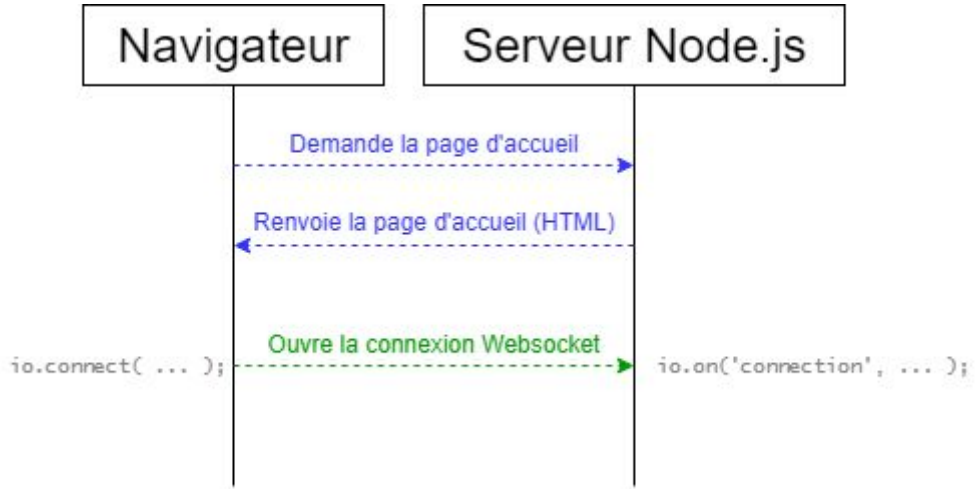
Il faut alors l'inclure dans la page web comme toute librairie client classique :

```
<script src="/socket.io/socket.io.js"></script>
```

Afin qu'un client instancie une connexion Websocket lorsqu'il charge la page, il faut appeler ma méthode `io.connect` vers l'adresse du serveur web socket :

```
// Initialisation de la connexion au serveur Websocket
const socket = io.connect('http://localhost:9000');
// "socket" est un objet représentant ce socket client unique
```

L’ouverture d’un serveur Websocket peut s’illustrer de la façon suivante :



À vous de jouer.

Installez socket.io via npm : `npm install --save socket.io`

Votre application Express écoute sur le port 9000.
Procédez aux modifications nécessaires pour y coupler socket.io et faire en sorte que chaque client demandant la page d’accueil ouvre un socket sur le serveur, qui logguera pour l’instant un simple message.

Observez le résultat dans votre terminal :

```
➜  Chat node index.js
✓ Le serveur écoute sur le port 9000
Client 81qdi5-Gmpet2_T-AAAA is connected via WebSockets
```