# Tutoriel

WebSocketServer

Version : 1.0

Révision : 1.0

Etat : Version 1.0

Equipe : Dream Team

Année : 2021 – 2022

Responsable du document : Timothée GIRARD

| Date | Action | Auteur | Version | Révision |
|---|---|---|---|---|
| 20/07/2021 | Création | Rayan L. | 1.0 | 1.0 |
| 23/08/2021 | Création | Timothée G. | 1.0 | 1.0 |

*Tableau 1 - Table des versions*

# Tables des tableaux



# Tables des figures

**Aucune entrée de table d'illustration n'a été trouvée.**

# Table des matières

# 1 Introduction

L'objectif de ce présent document est de comprendre comment le serveur Web Socket a été créé et son principe de fonctionnement. Il fonctionne grâce à node.js et permet de pourvoir communiquer à distance via les technologies Web. Le projet est disponible sur GitHub à l'adresse suivante : https://github.com/Timot97G/WebSocketServer.

# 2 Définition, acronymes et abréviations

Voici les termes et abréviations nécessaires à la compréhension du dossier de spécification :

| Terme ou abréviation | Signification |
|---|---|
| Uniform Resource Locator | Adresse d'un site ou d'une page hypertexte sur Internet |

*Tableau 2 - Sigles pour la documentation*

# 3 Références

Voici un tableau récapitulatif des documents utilisés pour le dossier de spécification ainsi que les liens permettant d'accéder aux fichiers :

| Référence | Description | Version | Auteur | Date |
|---|---|---|---|---|
| | | | | |

*Tableau 3 - Références documents*

# 4 Les étapes

## 4.1 Coté serveur

### 4.1.1 Etape 1 : Installation de node.js

Dans un premier temps, il faut télécharger et installer node.js ( https://nodejs.org/fr/). Pour savoir si tout est correctement installé utiliser les commandes : ***node -v*** ou ***npm -v***.

### 4.1.2 Etape 2 : Initialisation de node.js

Ensuite, on initialise node.js à l'aide de l'invité de commande à l'emplacement du dossier. On entre les commandes suivantes :

***npm install -g npm*** (pour mettre à jour npm)

***npm install express*** (récupérer l'essentiel de node_module)

***npm install ws*** (récupérer la librairie web socket)

### 4.1.3 Etape 3 : Création du serveur en JavaScript

Après cela, on va créer un fichier qu'on appellera « server.js » avec le code suivant :

```
const express =require('express');
const http = require('http');
const WebSocket = require('ws');

const port = 8100;
const server = http.createServer(express);
const wss = new WebSocket.Server({ server })

wss.on('connection',function connection(ws){

   ws.on('close', ()=>{
      //Do something
   })
   ws.on('message', function incoming(data){
      //Do something
   })
})

server.listen(port,function(){
    console.log(`Server is listen on ${port}`)
})
```

*Explication :*

L'objet « wss » représente le serveur, il contient l'ensemble des clients qui y sont connectés, « ws » est l'instance (client) actuelle. Lorsqu'un client se connecte au serveur, les listeners « close » et « message » lui sont attribués. On peut aussi lui attribuer des variables en faisant par exemple :

```
ws.var = 'valeur'
```

#### 4.1.4 Etape 4 : Transition des messages

Quand le client envoie un message vers le serveur et que l'on souhaite que celui-ci le redistribue aux autres clients, on peut procéder de la façon suivante :

```
ws.on('message', function incoming(data){

   //Boucle sur tous les clients le l'objet wss
   wss.clients.forEach(function each(client) {

      //vérifie si le client de la boucle est prêt à (envoyer/recevoir) des
infos
      if (client.readyState === WebSocket.OPEN) {

         //vérifie si le client n'est pas le même qui a envoyé le message
         if (client !== ws) {

             //le server envoie un message au client correspondant
             client.send(data)
         }
      }
   }
}
```

#### 4.1.5 Etape 5 : Lancement du serveur

Pour finir, il faut lancer le server à l'aide de la commande ***node server.js*** (dans l'invité de commande à la racine du dossier). Si vous avez suivi la procédure, vous devriez obtenir le message *« Server is listen on 8100 »*, cela signifie que le serveur est bien lancé.

### 4.2 Côté client

#### 4.2.1 Etape 6 : Connexion du client au serveur

Pour qu'un client se connecte au serveur Web Socket, il suffit d'une ligne :

```
let wss = new WebSocket('ws://localhost:8100');
```

Pour un serveur hébergé sur internet, il suffit d'entrer l'url associée au script (sans devoir spécifier le port) *[le port à utiliser peut être défini par l'hébergeur]*.

*Exemple* : Si l'url pour accéder au fichier est définie sur https://monsite.com/serv, il faudra mettre wss://monsite.com/serv.

*Astuce* : Si le serveur est correctement lancé et que vous vous rendez sur l'url correspondant au fichier (https://monsite.com/serv), alors la page devrait charger indéfiniment.

➔ Informations : ws = non sécurisé, wss = sécurisé, le localhost utilise le ws.

Les méthodes et attributs les plus utiles pour l'utilisation du Web Socket sont les suivant :

```
wss.onopen = ()=>{ .. }
wss.onerror = (e)=>{ .. }
wss.onclose = ()=>{ .. }
wss.onmessage = ({data})=>{ .. }

wss.send("message");
wss.close();
```