

# Codelab: Utiliser "ws" pour créer un serveur WebSocket

## Objectifs

Le protocole WebSocket permet une communication bidirectionnelle entre un client et un serveur. Les WebSockets sont couramment utilisés pour créer des applications Web en temps réel, telles que des clients de messagerie instantanée.

Dans ce codelab, nous allons utiliser le module tiers **ws** pour créer un serveur WebSocket avec lequel nous pouvons interagir via notre navigateur.

## Créer un nouveau projet

Commencez par créer un répertoire nommé **websocket-server** contenant deux fichiers, l'un nommé **client.js** et l'autre nommé **server.js** :

```
$ mkdir websocket-server
$ cd websocket-server
$ touch client.js
$ touch server.js
```

Aussi, pour notre client, créons un répertoire public contenant un fichier nommé **index.html** :

```
$ mkdir public
$ touch public/index.html
```

Comme nous allons utiliser un module npm tiers, nous devons également initialiser notre projet :

```
$ npm init --yes
```

## Créer le serveur et le client

Dans cet codelab, nous allons créer un serveur WebSocket et un client et envoyer des messages entre les deux.

Commencez par installer le module **ws** :


```
$ npm install ws
```

Importez le module **ws** dans **server.js** :



```
const WebSocket = require("ws");
```

Maintenant, nous pouvons définir notre **WebSocketServer**, y compris sur quel port il doit être accessible :



```
const WebSocketServer = new WebSocket.Server({  
  port: 3000,  
});
```

Nous devons écouter les connexions et les messages vers notre **WebSocketServer** :

```
WebSocketServer.on("connection", (socket) => {
  socket.on("message", (msg) => {
    console.log("Reçu:", msg.toString());
    if (msg.toString() === "Bonjour BUT") socket.send("Helloooo!");
  })
});
```

Maintenant, créons notre client. Ajoutez ce qui suit à **client.js** :

```
const fs = require("fs");
const http = require("http");
const index = fs.readFileSync("public/index.html");
const server = http.createServer((req, res) => {
  res.setHeader("Content-Type", "text/html");
  res.end(index);
});
server.listen(8080);
```

Ouvrez **index.html** et ajoutez ce qui suit :

```

<html>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<body>
<h1>Communiquer avec WebSockets</h1>
<input id="msg" /><button id="send">Envoyer</button>
<div id="output"></div>
<script>
  const ws = new WebSocket("ws://localhost:3000");
  const output = document.getElementById("output");
  const send = document.getElementById("send");
  send.addEventListener("click", () => {
    const msg = document.getElementById("msg").value;
    ws.send(msg);
    output.innerHTML += log("Envoyé", msg);
  });
  function log(event, msg) {
    return "<p>" + event + ": " + msg + "</p>";
  }
  ws.onmessage = function (e) {
    output.innerHTML += log("Reçu", e.data);
  };
  ws.onclose = function (e) {
    output.innerHTML += log("Déconnecté", e.code);
  };
  ws.onerror = function (e) {
    output.innerHTML += log("Erreur", e.data);
  };
</script>
</body>
</html>

```

Maintenant, démarrez votre serveur dans une fenêtre Terminal et votre client dans une seconde fenêtre Terminal :

```

$ node server.js
$ node client.js

```

Accédez à **http://localhost:8080** dans votre navigateur et vous devriez voir une simple zone de saisie avec un bouton **Envoyer**. Tapez **Bonjour BUT** dans la zone de saisie et cliquez sur **Envoyer**. Le serveur WebSocket devrait répondre avec **Helloooo!**. Si nous jetons un coup d'œil à la fenêtre Terminal où nous exécutons notre serveur, nous devrions voir que le serveur a reçu le message. Cela signifie que nous avons maintenant un client et un serveur communiquant via WebSockets.

← → ↻ ⓘ localhost:8080

# Communiquer avec WebSockets

Bonjour BUT

Envoyé: Bonjour BUT

Reçu: Helloooo!

Dans cet exemple, nous avons utilisé le module **ws** pour définir un serveur **WebSocket**. Nous avons ensuite enregistré un écouteur pour l'événement de connexion. La fonction passée est exécutée à chaque nouvelle connexion au WebSocket. Dans la fonction de rappel d'événement de connexion, nous avons enregistré un écouteur imbriqué pour l'événement de message (**message**), qui est exécuté à chaque fois qu'un message est reçu.

Pour notre client, nous avons défini un serveur HTTP standard pour servir notre fichier **index.html**. Notre fichier **index.html** contient du JavaScript qui est exécuté dans le navigateur. Dans ce JavaScript, nous avons créé une connexion à notre serveur **WebSocket**, fournissant le point de terminaison que l'objet **ws** écoute : **const ws = new WebSocket("ws://localhost:3000");**

Pour envoyer un message à notre serveur **WebSocket**, nous appelons simplement **send** sur l'objet **ws** avec **ws.send(msg)**.

Nous avons enveloppé le **ws.send(msg)** dans un écouteur d'événement. L'écouteur d'événement écoutait l'événement "**click**" sur le bouton "**Envoyer**", ce qui signifie que nous envoyions le message au WebSocket lorsque le bouton était cliqué.

Dans notre script dans **index.html**, nous avons enregistré les fonctions d'écouteur d'événements sur notre **WebSocket**, y compris les écouteurs d'événements **onmessage**, **onclose** et **onerror**. Ces fonctions s'exécutent sur leurs événements respectifs. Par exemple, la fonction d'écoute d'événement **onmessage** s'exécute lorsque notre WebSocket reçoit un message. Nous utilisons ces écouteurs d'événements pour afficher les messages de sortie dans notre page Web.


## WebSocket client

Maintenant, nous avons appris comment nous pouvons communiquer entre un navigateur et un serveur à l'aide de WebSockets. Mais il est également possible de créer un client WebSocket dans Node.js, permettant à deux programmes Node.js de communiquer via WebSockets.

Commencez par créer un nouveau fichier dans notre répertoire **websocket-server**, nommé **node-client.js** :

```
$ touch node-client.js
```

Ouvrez **node-client.js** et importez le module **ws** et créez un nouvel objet **WebSocket** configuré pour pointer vers le serveur **WebSocket** que nous avons déjà créé.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The terminal displays two lines of JavaScript code: 

```
const WebSocket = require("ws");  
const ws = new WebSocket("ws://localhost:3000");
```

```
const WebSocket = require("ws");  
const ws = new WebSocket("ws://localhost:3000");
```

Maintenant, nous allons configurer des écouteurs sur nos sockets. Nous ajouterons des écouteurs pour les événements d'ouverture, de fermeture et de message :

```
ws.on("open", () => {
  console.log("Connecté");
});
ws.on("close", () => {
  console.log("Déconnecté");
});
ws.on("message", (message) => {
  console.log("Reçu:", message);
});
```

Envoyons maintenant le message "Bonjour, c'est node-client.js" au serveur WebSocket toutes les 3 secondes. Nous utiliserons la fonction **setInterval()** pour y parvenir :

```
setInterval(() => {
  ws.send("Bonjour, c'est node-client.js");
}, 3000);
```

Démarrez le serveur WebSocket et votre client basé sur Node.js dans des fenêtres Terminal distinctes :

```
$ node server.js
```

```
$ node node-client.js
```

```
(base) josephazar@Josephs-MBP-2 websocket-server % node server.js
Reçu: Bonjour, c'est node-client.js
Reçu: Bonjour, c'est node-client.js
Reçu: Bonjour, c'est node-client.js
Reçu: Bonjour, c'est node-client.js
□
```

```
(base) josephazar@Josephs-MBP-2 websocket-server % node node-client.js
Connecté
□
```

Vous avez maintenant créé une communication WebSocket entre deux programmes Node.js.