MEMENTO Méthode de création d'une communication client-serveur de type point socket.io de départ Création d'un dépot sur github "la_parole_a_tous" Création du server Création socket.io côté client socket.io (on va intégrer une Cloner le projet et \$ git clone <adresseDepot>.git gestion http pour prévenir des aller dans le \$ cd a_parole_a_tous éventuels pb de repertoire nctionnement avec socket.io (le plan B en qq sorte) On se repostionne dans le repertoire \$ cd . racine du projet la_parole_a_tous Création dans \$ mkdir server l'arborescence de notre \$ cd server \$ mkdir client projet d'un Création d'un dossier \$ cd client répertoire dédié au server dédié au back création du projet Node \$ npm init -y (création du fichier package.json) \$ touch \$ touch index.html Création de la page Création du script qui server.js va piloter le serveur index.html Installation des outils (dépendances) utilisées par le server.js \$ npm install --save-dev nodemon Récupération des ml lang="fr"> \$ npm install socket.io express socket fonctions qui nous /*Dépendances du script*/ -nodemon Récupération du script interressent dans les exposé par le serveur -express import express from "express"; /*Import de tout ce qui est dispo dans le module express.*/ dépendances pour notre socket.io script ta http-equiv="X-UA-Compatible" content="IE=edge"> import{createServer}from"http"; /*Import de la fonction createServer dans le module http*/ pour accéder aux fonctionnalités de socket.io import{Server}from"socket.io"; /* import de la fonction Server dans le module socket.io*/ Configuration du fichier de on rajoute dans package.json: configuration du projet la propriété "start" à l'objet script: (package.json) script src="/socket.io/socket.io.js"></script> "start":"nodemon server.js" - Affectation de la commande server.js Création d'un d'exécution du script à l'option et également, raccourci vers le 'start' de npm une propriété type à la prorpiété dossier dédié à nos app.use(express.static("public")); globale: ressources statiques - renseignement du type module "type"="module" pour éviter de passer par des fichiers mjs On signale à notre app qu'on veut utiliser scriptJavascript server.js express. On invoque déclenchement de la la fonction express() connexion avec le du module express const app=express(); serveur const socket = io(); qui nous renvoie un Création du server objet que nous socket.io nommons app script Javascript Affectation des server.js Création du serveur différentes fonctions http - qui va utiliser callbacks aux différents express - à l'aide de la const httpServer=createServer(app); socket.on('my-event-b',()=>{// ...}); évènements fonction createServer reçus du module http script Javascript server.js Création d'un deuxieme envoi d'évènements serveur socket.io, qui est une const io=newServer(httpServer,{}); surcouche du serveur http, qui // socket.emit('my-event-a'); va utiliser le protocole websocket. Notre serveur socket sera représenté par l'objet io Création du socket.io côté fin client Affectation et définition server.js des différentes fonctions callbacks qui réagiront aux évènements émis io.on("connection",(socket)=>{//code qui v}a s'executer à chaque évènement 'connection' détecté}); \$ cd .. depuis le front. \$ git add . On va commencer par Premier commit du \$git commit -m "structure projet OK" l'évènement 'connection' projet \$ git push origin <nom_branche> Affectation du numéro de server.js port sur lequel le serveur va écouter puis réagir // écoute du port 3000 lorsque des évènements seront io.listen(3000); détectés