IUT Dijon - Licence Professionnelle « Métiers du Numérique »
TD NodeJS – Jérémie Brizard
jeremie.brizard@gmail.com

TD NodeJS : chat collaboratif

1. Création d'un compte GitHub

Allez sur github.com et créez-vous un compte si vous n'en avez pas déjà un. Si vous travaillez en groupe, vous n'avez besoin que d'un seul compte. Notez bien le mot de passe car vous en aurez besoin à chaque fois que vous voudrez publier du code (« commit »).

Une fois que votre compte est créé, donnez moi votre nom d'utilisateur pour que je puisse vous ajouter comme collaborateur du projet.

Le projet est visible à cette adresse : https://github.com/jbrizard/ddim_chat_2022

Prenez le temps de lire le fichier « Conventions de code » car vous devrez respecter ces conventions pour que votre code soit accepté dans le projet !

2. Mise en place du projet

Créez un dossier (dans vos documents ou autre) pour accueillir les futurs fichiers du projet.

Ouvrez un invite de commande DOS (cmd) et naviguez jusqu'au dossier que vous venez de créer.

Entrez la commande suivante pour obtenir un clône du projet Git :

```
git clone https://github.com/jbrizard/ddim_chat_2022.git
```

Vous devriez voir apparaître un dossier « ddim_chat » dans votre dossier conteneur. Prenez le temps de bien explorer les différents répertoires et fichiers du projet avant de vous lancer.

3. Créer votre branche Git

Choisissez un nom pour votre groupe. Ce nom devra être normalisé sous cette forme : nom-du-groupe (lettres minuscules et tirets uniquement).

Toujours dans l'invite de commande, tapez la commande suivante :

`git checkout -b nom-du-groupe` *(mettez le nom de votre groupe bien sûr)*

Vous avez maintenant votre propre branche dans laquelle vous allez pouvoir coder sans impacter le travail des autres groupes.

4. Premier commit

Ouvrez le fichier « AUTHORS » et dans la rubrique « Contributeurs actifs », saisissez les noms des participants de votre groupe **sur une seule ligne**, séparés par des virgules, ainsi que le nom de votre branche entre parenthèses, par exemple :

```
Jérémie Brizard (team-prof)
```

Puis refermez le fichier et rouvrez l'invite de commande. Tapez les commandes suivantes :

```
git add .
git commit -a -m "Premier commit du groupe nom-du-groupe"
git push origin nom-du-groupe
```

Remarque : vous utiliserez toujours cette syntaxe pour envoyer votre code une fois terminé (avec un autre message)

5. Où écrire son code ?

Lorsque vous avez choisi une fonctionnalité à créer, commencez par réfléchir à quelle partie du code va aller côté client et quelle partie va se faire côté serveur.

Ensuite, essayez d'estimer à quels endroits vous allez devoir intervenir dans le code initial, et de quelles variables vous aurez besoin.

Dans la mesure du possible, vous devez « **encapsuler** » votre code dans des fonctions, qui seront elles-mêmes définies dans un **fichier de module** séparé du code principal. Ensuite, vous modifierez le code initial uniquement pour appeler vos fonctions en passant les paramètres nécessaires.

Vérifiez à chaque fois que vous intervenez dans le code initial que le fonctionnement du chat n'est pas impacté.

Voici un exemple :

server.js (code initial)

```
43.     // Transmission d'un message
44.     socket.on('message', function(message)
45.     {
46.         // Échappe les caractères spéciaux
47.         message = ent.encode(message);
48.
49.         // Transmet le message en "broadcast" (envoie à tous les utilisateurs…
50.         socket.broadcast.emit('new_message', {user:socket.user, message:message});
51.
52.         // Module Daffy
53.         daffy.handleDaffy(io, message);
54.     });
```

modules/daffy.js (module)

```
5.  /**
6.   * Lorsqu'on appelle Daffy, il répond...
7.   */
8.  function handleDaffy(io, message)
9.  {
10.     if (message.startsWith('daffy ?'))
11.     {
12.         io.sockets.emit('new_message',
13.         {
14.             user:'DaffyDuck',
15.             message:'<span class="daffy">Coin Coin !</span>'
16.         });
17.     }
18. }
```

De cette manière, vous limitez au maximum votre impact sur le code initial, et l'assemblage sera beaucoup plus facile à la fin.