Rapport TP NodeJS

Téo BRYER

21/12/2018

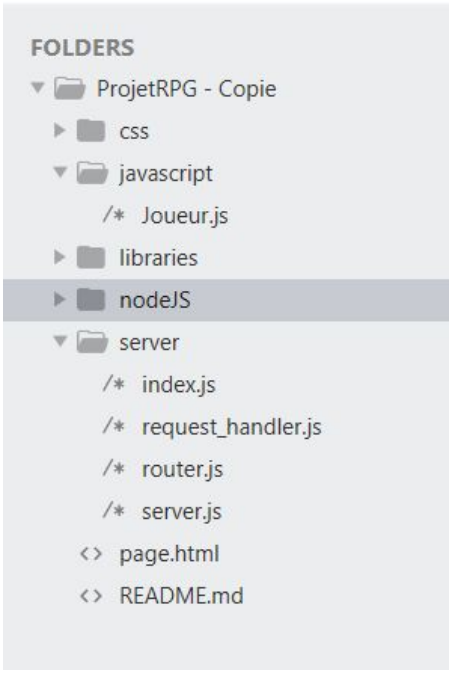
Version Node JS

Version 10.13

Description du projet

Dans ce TP, nous devons réaliser un RPG (sur le thème de votre choix), avec une architecture client/serveur

Hiérarchie des archives



Là où j'en suis rendu

Je suis rendu à la dernière question du node JS 4.4 je n'ai juste pas appliqué le multiplicateur.

Les classes

Classe Monstre

permet d'instancier un monstre avec un nom des hp et des dégâts

Classe Arme

permet d'instancier une arme avec un nom et des dégâts

Classe Armure

permet d'instancier une armure avec un nom et une valeur

Classe Joueur

permets d'instancier un joueur avec un nom, des points de vies, de l'agilité, de l'intelligence, (un inventaire et une position x et y)

REQUESTHANDLER

load

Fonctionnalité : charger les données du joueur du fichier JSON (après un save)

Nom de la fonction : load(request, response)

Description : permet de renvoyer au client les données du joueur sauvegardé ainsi que instancier le joueur du serveur avec les données du fichier JSON

Paramètres : aucun

Sorties : renvoi une réponse(le joueur) au load du client

Logique = récupère le joueur sur le fichier le JSON l'envoyer au client et le parser pour récupérer le joueur sur la variable joueur du serveur

save

Fonctionnalité : sauvegarder les données du joueur dans un fichier JSON Nom de la fonction : save (request, response)

Nom de la fonction : load(request, response)

Description : permet de sauvegarder le joueur dans un fichier JSON pour ensuite le load

Paramètres : aucun

Sorties : aucun

createPerso

Fonctionnalité : créer un personnage sur le serveur

Nom de la fonction : createPerso(request, response)

Description : permet de créer un joueur grâce aux données du formulaires remplies par l'utilisateur en récupérant les données via l'URL

coffreEvent

permet de gérer l'événement d'un coffre, renvoi le texte associé à cet évènement

armureEvent

permet de gérer l'événement d'une armure, renvoi le texte associé à cet évènement

piegeEvent

permet de gérer l'événement d'un piège, renvoi le texte associé à cet évènement

monstreEvent

permet de gérer l'événement d'un monstre, renvoi le texte associé à cet évènement (et lance la métode combat())

evenement

permet de générer un événement aléatoire (parmis les événements précédents)

go

permet de gérer le déplacement du joueur en récupérant la direction du joueur en via l'url et permet de lancer la méthode évènement.

combat

permet de gérer un combat et savoir qui meurt.

JOUEUR JS

tableau()

permet l'affichage des données du joueurs (seulement celles éditables par le joueur lors de la création)

afficherDeplacement()

permet d'afficher le déplacements et les évènements du joueur

go(id)

permet de déléguer la méthode pour se déplacer au serveur en donnant en paramètre du get l'id de déplacement (nord sud est ouest)

load()

permet de déléguer au load du serveur pour récupérer le joueur sauvegardé préalablement

save()

permet de déléguer la sauvegarde au serveur et enregistrer le joueur dans un fichier JSON.

createPerso()

permet de déléguer la création au serveur