Space Storm

Serveur NodeJS WebSocket

26/11/2015

# JavaScript (CoffeeScript)

## Langage principal du serveur Space Storm

Le langage utilisé pour le serveur Space Storm est le JavaScript compilé en CoffeeScript.

CoffeeScript est un [langage de programmation](https://fr.wikipedia.org/wiki/Langage_de_programmation), qui se [compile](https://fr.wikipedia.org/wiki/Compilateur) en [JavaScript](https://fr.wikipedia.org/wiki/JavaScript). Le langage ajoute du [sucre syntaxique](https://fr.wikipedia.org/wiki/Sucre_syntaxique) inspiré par [Python](https://fr.wikipedia.org/wiki/Python_(langage)), [Ruby](https://fr.wikipedia.org/wiki/Ruby) et[Haskell](https://fr.wikipedia.org/wiki/Haskell) afin d'améliorer la brièveté et la lisibilité du JavaScript, tout en ajoutant des fonctionnalités comme le [filtrage par motif](https://fr.wikipedia.org/wiki/Filtrage_par_motif) ou les [listes en compréhension](https://fr.wikipedia.org/wiki/Liste_en_compr%C3%A9hension).

Le résultat est compilé de façon prévisible en JavaScript, et les programmes peuvent être écrits avec moins de code (typiquement un tiers de lignes en moins) sans effet sur la vitesse d'exécution. Depuis le [16](https://fr.wikipedia.org/wiki/16_mars) [mars](https://fr.wikipedia.org/wiki/Mars_2011) [2011](https://fr.wikipedia.org/wiki/2011), CoffeeScript est dans la liste des projets les plus populaires de [GitHub](https://fr.wikipedia.org/wiki/GitHub).

Kingoloto possède deux développeurs formé sur ce langage (CoffeeScript). Qui servira sur le serveur NodeJS. Le module ‘ws’ permet au serveur de s’orienter vers du WebSocket. Le serveur est initialisé dans la Classe Main.coffee ainsi que c’est paramètre.

|  |  |  |
| --- | --- | --- |
|  |  |  |

### Tout les Classes sont déclaré en ‘global’ pour qu’ils soient accessibles partout.

Liste des Classes :

* app.coffee (Initialisation du serveur WebSocket)
* bonusShop.coffee (Classe de gestion du cashShop)
* Ennemi.coffee (Classe de gestion des Ennemi)
* Exp.coffee (Classe de gestion de l’expérience)
* Game.coffee (Classe d’extension des levels)
* gestToken.coffee (Classe de sécurité)
* player (Classe des players)
* Score.coffee (Classe de gestion du score global du serveur)
* Levelone.coffee (Classe du premier niveau)
* levelOneBoss.coffee (Classe Boss du premier niveau)
* leveltwo.coffee (Class du deuxième niveau)

**app.coffee :**

**constructor :** Fonction d’initialisation.

**StartTCP**: Fonction de démarrage du serveur WebSocket (sous le port 3000).

**setupDatabase(base)**: Fonction d’initialisation de la variable de connexion MySQL. Puis connecte la base de donnée (base).

**getConnection :** Fonction qui retourne la connexion MySQL

Apres un Timeout de 200 millisecondes, la classe est initialisé ainsi que la classe Score.

**bonusShop :**

**constructor(id) :** Retourne les nombres de bonus acheté ou gagné par le joueur (id).

**Ennemi:**

**constructor :** Fonction d’initialisation.

**getRandom(pourcentage) :** (pourcentage) permettra de définir la chance que l’ennemi soit retourné.

**generate :** Retourne les ennemies charger dans la base de donnée avec leurs armes.

**getEnnemi (id):** retourne l’ennemi par (id).

**Exp:**

**constructor (id):** Retourne les données du joueur en Json.

**Game:**

**getXY:** Retourne une position X et Y.

**sendAllPlayer (json):** Envoie a tout les joueurs le packet (json).

**getfinalScore:** Retourne le score final.

**setfinalScore:** Initialise un score final.

**getRandomNumber:** Retourne un nombre au hasard.

**gestToken:**

**constructor (token):** Rends le (token) écoutable sur toute la classe.

**getId (id):** Retourne le token du joueur (id).

**Player:**

**Constructor (@socket, @id):** Initialise un joueur.

**Save :** Sauvegarde les données du joueur.

**Close (@socket) :** Fonction qui se lance lorsque le joueur se déconnecte.

**text (@socket, @id) :** Fonction qui reçoit les paquets du Client.

**Score:**

**Constructor :** Initialise le scores des ennemis.

**initScore:** Initialise le score global.

**SuprScore (nb):** Enlève (nb) au score global.

**addScrore (type):** Ajoute l’expérience de l’ennemi (type) au score global.

**getExpEnnemi (type):** Retourne l’expérience de l’ennemi (type).

**getScoreEnnemi:** Retourne les données expérience des ennemis.

**getScore:** Retourne le score.

**Levelone extends Game:**

**Constructor :** Initialise le score du level.

**Boss :** Lance le Boss lorsque le score du level est atteint.

**Start :** Gestion du level.

**leveloneboos extends Game:**

**Constructor :** Initialise le score du level.

**Leveltwo :** Lance le deuxième level lorsque le score du level est atteint.

**Start :** Gestion du level.

**Leveltwo extends Game:**

**Constructor :** Initialise le score du level.

**Boss :** Lance le Boss lorsque le score du level est atteint.

**Start :** Gestion du level.