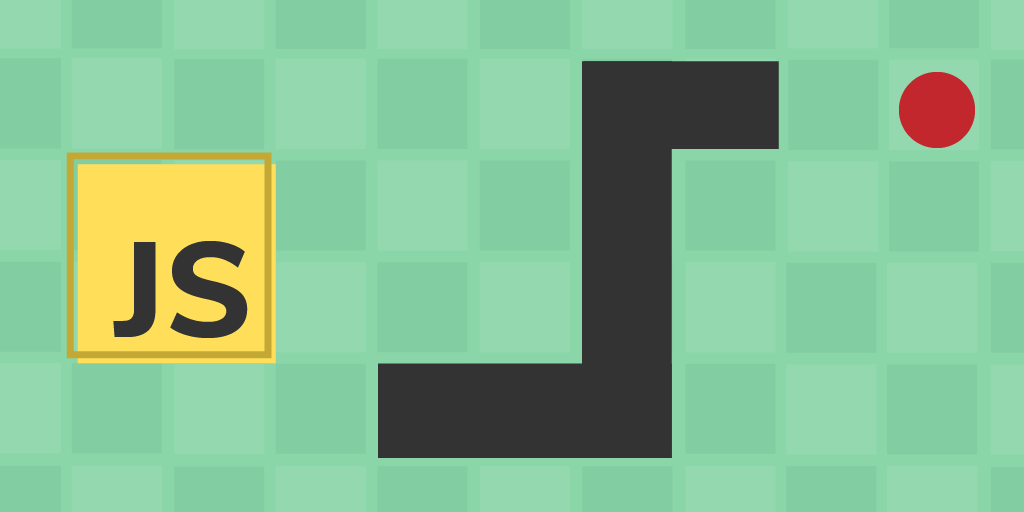
Projet Bulles



**Rapport de Projet**

Zeqiri Amir – CID2A

ETML, Sébeillon – N512A

40 Périodes

Maître : M. Chenaux

Table des matières

[1 Description du projet dans son ensemble 3](#_Toc153895086)

[1.1 Titre 3](#_Toc153895087)

[1.2 Sujet 3](#_Toc153895088)

[2 Support de cours 3](#_Toc153895089)

[2.1 Importation de la machine virtuelle 3](#_Toc153895090)

[2.2 Installation des prérequis 3](#_Toc153895091)

[2.3 Codage du Snake en JS 4](#_Toc153895092)

[2.3.1 Utiliser const et let mais jamais var 6](#_Toc153895093)

[2.3.2 Créer des classes (Snake, Apple, etc) 6](#_Toc153895094)

[2.3.3 Utiliser les fonctions fléchées (sauf dans le cas des méthodes d’une classe) 6](#_Toc153895095)

[2.3.4 Les modules : import / export 7](#_Toc153895096)

[2.3.5 Utiliser l’opérateur rest pour décomposer un tableau 7](#_Toc153895097)

# Description du projet dans son ensemble

## Titre

Snake en JS

## Sujet

Faire un réplica du jeu célèbre jeu Snake

# Support de cours

## Importation de la machine virtuelle

La machine virtuelle sur laquelle nous allons faire notre projet nous a été donné par le maître M. Chenaux. Nous avons dû l’importer sur Oracle VM VirtualBox afin de pouvoir l’ouvrir comme il se doit. La machine était toute configurée et Visual Studio Code était installé au préalable. Personnellement, j’ai installé l’extension Live Preview sur VS Code. C’est une extension Microsoft qui me permet d’avoir un rendu de mon code et de voir les changements de mon code en temps réel sur mon navigateur (localhost). La VM fonctionne sur Linux Ubuntu, c’est un changement pour moi car la grande majorité de mon temps passé sur un ordinateur est sur Windows. Malgré ça, c’est un OS où j’ai déjà pu travailler dessus et qui n’est pas compliqué à utiliser.

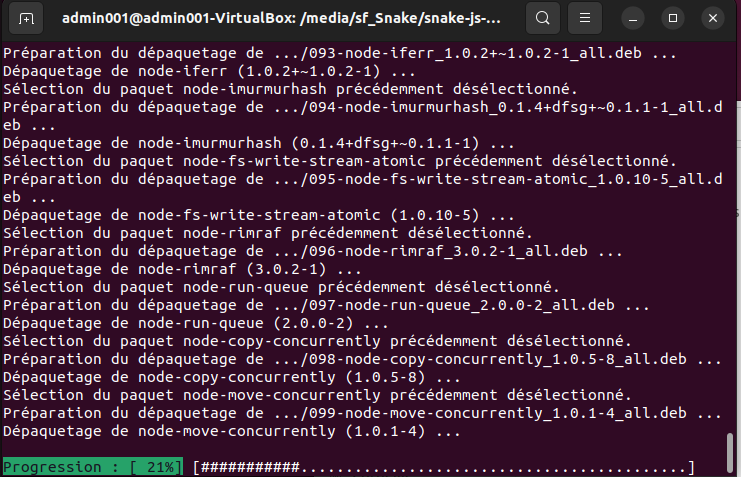
## Installation des prérequis

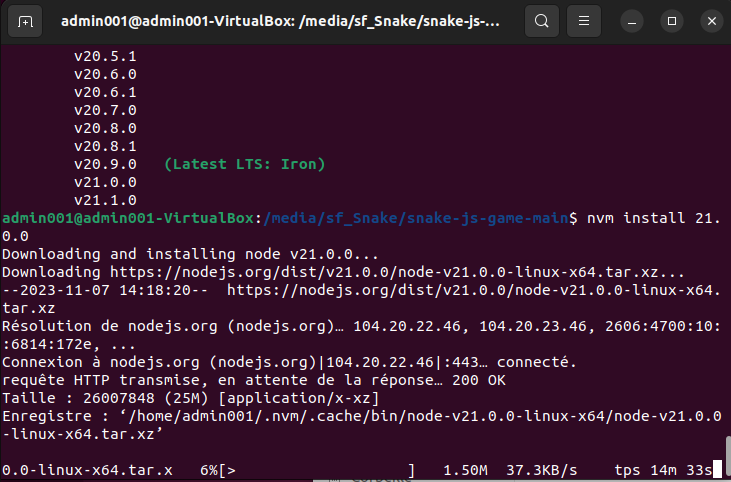
Avant de commencer quoi que ce soit, j’ai dû installer les dépendances JS (Node JS) pour le bon fonctionnement du projet.

Voici comment j’ai procédé :

1. Ouvrir le terminal dans le dossier où se trouve le snake
2. Ecrire "sudo apt install npm"
3. Ecrire "sudo apt update"
4. Ecrire "sudo wget –q0-"
5. Ecrire "source ~/.profile"
6. Ecrire "nvm –v"
7. Ecrire "nvm ls-remote"
8. Ecrire "node –v"

Et voici quelques photos prises durant la procédure :





## Codage du Snake en JS

Il est maintenant grand temps de commencer à coder mon Snake en JavaScript. Nous disposons d’une base pour commencer le projet, en effet, nous avons le fichier index en HTML, le fichier source en CSS, le fichier main en JavaScript et les différents packages qui je ne devrai normalement pas y toucher. La base qui nous a été donné ne fonctionnais pas parfaitement, le CSS n’était pas lié à l’HTML. J’ai simplement ajouté cette ligne dans le "head" de ma page index :

<link rel="stylesheet" href="css/style.css">.

Une fois ceci ajouté, j’avais ce rendu :



On peut voir que le canvas est là et que le CSS est bien relié. Je peux maintenant, de moi-même, coder le snake.

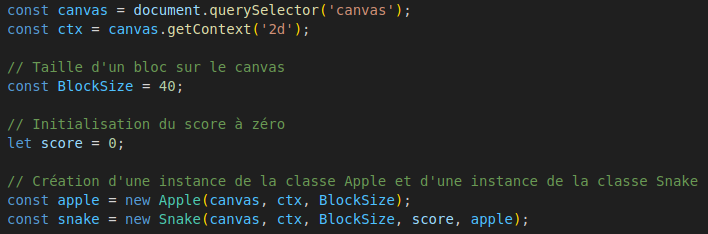
Je commence donc par mettre les bords du canvas en vert, comme la couleur du serpent et je fais en sorte à ce qu’il n’y ait pas de curseur sur le canvas afin que, quand je joue, cela ne me dérange pas.

Voici le rendu :



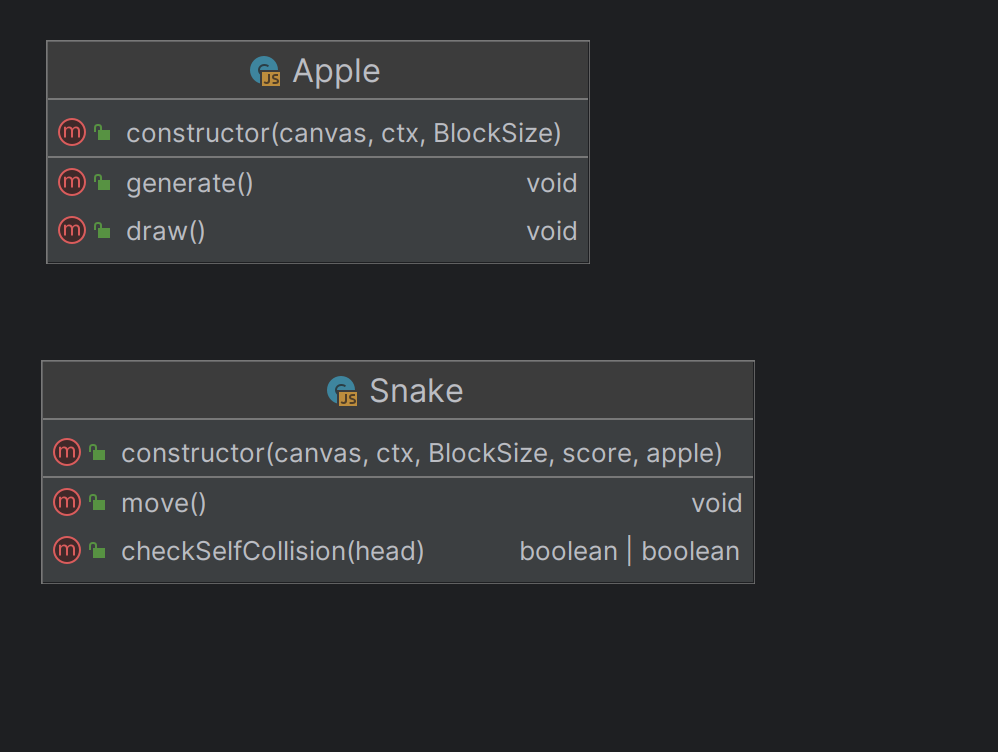
Ensuite, je vais me consacrer au sérieux, c’est-à-dire afficher un bloc vert qui sera la tête du serpent, et à chaque fois que le serpent mangera une pomme un bloc s’ajoutera à lui qui suivra sa trace.

### Utiliser const et let mais jamais var



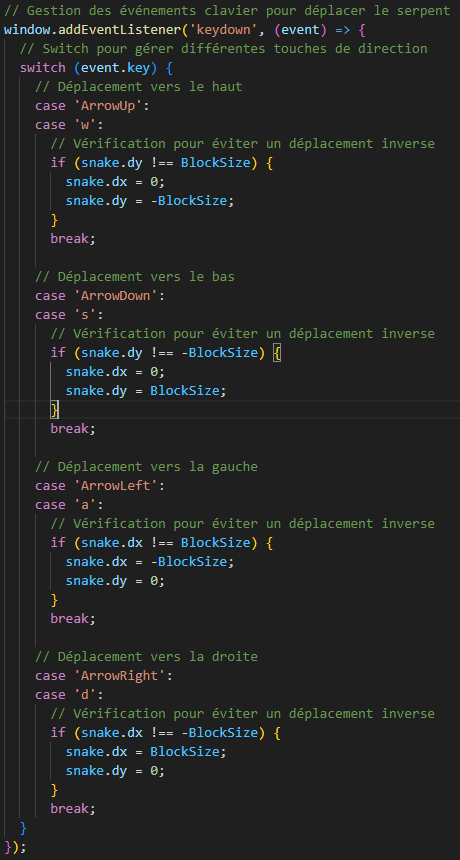
Ce printscreen montre les variables que j’utilise dans le main. On peut voir qu’aucune variable var n’est déclarée. 5 sont déclarés en const et une seule en let.

### Créer des classes (Snake, Apple, etc)



Sur ce diagramme de classes sont affichés, les différentes classes que j’utilise : "Apple" et "Snake"

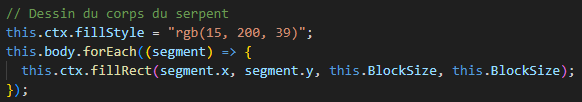
### Utiliser les fonctions fléchées (sauf dans le cas des méthodes d’une classe)

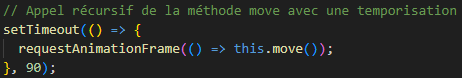


Sur cette capture de mon code, on voit la première fonction fléchée.

La première fonction fléchée fonctionne comme un gestionnaire d'événements pour écouter les touches du clavier. Elle est liée à l'événement 'keydown' sur l'objet window. Cette fonction permet de détecter quelle touche du clavier est enfoncée et ajuste ensuite la direction du serpent en conséquence. Elle gère les déplacements vers le haut, le bas, la gauche et la droite en fonction des touches fléchées et des touches 'W', 'S', 'A' et 'D'.

La deuxième fonction fléchée sert de callback dans la méthode requestAnimationFrame. Elle est appelée à chaque frame d'animation et elle déclenche la méthode move de l'instance de la classe Snake. Cela établit une boucle principale qui actualise la position du serpent à chaque frame, ça assure un mouvement fluide.

La troisième fonction fléchée, utilisée comme callback dans setTimeout, enveloppe un appel à requestAnimationFrame. Cette dernière utilise une autre fonction fléchée comme callback pour appeler récursivement la méthode move. Tout cela crée une sorte de boucle non synchronisée qui permet au serpent de se déplacer en continu, avec un petit délai entre chaque étape, ce qui donne une animation fluide.

Cette fonction fléchée est utilisée comme callback pour la méthode forEach appliquée à l'array this.body. Elle est utilisée pour dessiner chaque segment du corps du serpent sur le canvas en utilisant les coordonnées fournies par chaque élément de l'array.

### Les modules : import / export

Dans la première image, j’exporte la classe "Snake"

Dans la deuxième image, j’exporte la classe "Apple"

Et dans la dernière image, j’importe les différentes classes dans le main

### Utiliser l’opérateur rest pour décomposer un tableau

Je n’utilise aucun opérateur rest dans l’intégralité de mon code.