TypeScript

The objective of this practical session is to discover one of the many languages that compiles to Javascript. TypeScript is a programming language developed by Microsoft. Its particularity is to allow the developer to define types for variables. It is important to note that the underlying type system is dynamic and thus types are optional. It ensures the compatibility with existing Javascript libraries. In this project we will always specify the types of the variables. You can found documentation on TypeScript here: http://devdocs.io/typescript/

Ssssssnake

You are going to implement a version of the famous Snake game (http://playsnake.org/) with TypeScript. The base code for the project is on GitHub : https://github.com/gbecan/teaching-jxs-typescript. It contains the following files:

- *index.html* will contain the game. It is mainly composed of a *canvas* element for drawing the different elements of the game and everything needed to load the code that you will develop.
- *Main.ts* is the entry point of our application.
- *Game.ts* will manage the different elements of the game, their animation and ensure the rules.

The first step of the implementation of the game is to model the snake and its movements. The snake will move on a grid. The size of the grid is defined by *gridWidth* and *gridHeight* in the Game class. The size of each cell is defined by the parameter *gridSize* in the Game constructor.

Q1: Create a class SnakePart to represent the coordinates of each part of the snake's body.

Q2: Create a class Snake that contains an array of SnakePart. Modify also the SnakeGame class to initialize the snake with a body of length 3.

Q3: Create a method *draw* in the classes Snake, SnakePart and Game to draw the snake in the canvas.

Move move move !

Now you should have a snake appearing on your game but it is not moving. It is time to implement the user interaction to make it move.

Q4: Create an enumeration SnakeDirection to model the direction of the snake.

Q5: Add a listener to the event *keydown* in order to detect the user interaction and change the direction of the snake. With this listener, you can detect when the user presses on the arrows of its keyboard. For

information, the key codes for the arrows are between 37 and 40.

Q6: Add a method *move* for managing the movement of the snake. You will call it each time you update the game.

Q7: Implement the rules of the game in the *updateGame* of the SnakeGame class. As a reminder, the head of the snake must not touch the side of the game space or its own body.

Are you hungry?

The last step is to manage the appearance of food and the growth of the snake.

Q8: Create a class Food to represent the coordinates of the food to catch.

Q9: Add a method *relocate* that moves the food to a random position. You should take care of avoiding to relocate the food on the snake.

Q10: Add a method *grow* to the Snake class that adds a SnakePart to the snake. Modify the *updateGame* method in order to manage the growth of the snake each time it catches food.

Q11: Each time the snake eats food, the score of the player is increased by 1. Create an element in the page to display the score and keep it updated.

Going further

Q12: Create a new type of bonus food that increases the score by 5 instead of 1. This new kind of food will appear randomly and for a limited time. Use the inheritance capabilities of TypeScript to implement this new class.

Q13: Create several difficulty levels. For example, the beginner level could have a reduced speed and not take into account the collision of the snake with its own body. The expert level could have an increased speed and food could appear only for a limited time. Here again, you can use the inheritance proposed by TypeScript and also design patterns.