JavaScript Snake Game

# Files:

* index.html (web page contents)
* app.js (logic of the game)
* app.css (styling of the web app)

# index.html

* Basic html boilerplate.
* Import css stylesheet inside the head tag.
* Title within the head tag.
* Layout of the web page within the body tag.
* We are using canvas tag to design the game.
* Include the javascript file in the end, within the body tag

# app.js

* “use strict” gives an error when a variable is declared without specifying its type. This leads to a cleaner code.
* We declare all the variables that we will be needing in the game and set a default value.
* We define a function **initialize()** that initializes the game canvas onto the screen:
  + We select the corresponding html elements by id and store them in a variable.
  + We then create a game area and define its width and height, along with a single cell size.
  + Then we set the game to start on clicking the ‘start game’ button.
* We define a function **startGame()** that performs the following operations:
  + Sets the initial score to 0.
  + Sets the starting snake direction to right.
  + We store the game speed inputted by user and validate the input.
  + Snake is declared as an array and initialised with a single cell value by pushing the cellwidth.
  + We call the **createFood()** function that creates a food cell randomly on the canvas.
  + Lastly, we reset the timer and then set the timer according to the game speed.
* We define a function **createFood()** that creates the food at random locations on the canvas:
  + snakeFood is a javascript object that has x and y names having values for width and height respectively.
  + Width and height are kept random at every function call to create Food randomly on canvas.
* We define a function **createGameArea()** that performs the following operations:
  + We set the snake x and y axis to initial values.
  + Then we fill the gameArea with white color and border as dark gray.
  + We then adjust snake’s X and Y positions according to its direction
  + Then we check for abnormal values of snake’s x and y axes:
    - if true:
      * We call **writeScore()** function that shows up the score on canvas.
      * Reset the timer.
      * Enable the game start button.
      * And stop the game.
  + We also check if snake encounters food:
    - if true:
      * We add that food to snake size.
      * Create a new food on the canvas.
    - Else:
      * We pop a cell from snake’s size when it moves forward and assign it to newHead.
  + newHead is added to beginning of the snake array.
  + Then we loop through the snake array and create squares by calling **createSquare()** function according to its length
  + We also create a square for snake food by calling the function.
* We define a function **Control(x, y, array)** that checks if the snake hits itself and returns true if it does, false if it doesn’t.
* We define a function **writeScore()** that prints user score on the screen when the game ends:
  + We set the font size and family.
  + Then, we set the font color.
  + After that, we set the game area to show the text on the canvas.
* We then define a function **createSquare(x, y)** that creates a square for the given x and y axes:
  + We define the color of the shape.
  + Then we define the cell dimensions.
* Lastly, we define the function **changeDirection(e)** that changes the direction of the snake:
  + We first take the keypress event and store the key value in a variable.
  + Every key on the keyboard is assigned a number.
  + We identify which key is pressed by checking its number and then move the snake in that direction.
* Then we set the events to the screen:
  + Whenever a key is pressed, **changeDirection()** function is called
  + **initialize()** function runs whenever the windows loads.

# app.css

* define css styles for body, which includes background, font styles, and basic layout.
* define the style for input element
* define the look and feel of canvas
* define the position of simplilearn logo