Snake Game

Tabla de Contenido

[1 Environment Setup 2](#_Toc96079642)

[1.1 Ingresa a tu cuenta de stackblitz 2](#_Toc96079643)

[2 index.html 2](#_Toc96079644)

[2.1 Pre-requisites 2](#_Toc96079645)

[3 style.css 2](#_Toc96079646)

[3.1 Style.css 2](#_Toc96079647)

[4 game.js 3](#_Toc96079648)

[4.1 Import de SNAKE\_SPEED, update as updateSnake, draw as drawSnake 3](#_Toc96079649)

[4.2 Import outsideGrid 3](#_Toc96079650)

[4.3 Import update as updateFood, draw as drawFood 3](#_Toc96079651)

[4.4 Declara una variable let llamada lastRenderTime y gameOver 3](#_Toc96079652)

[4.5 Declara una constante llamada gameBoard 3](#_Toc96079653)

[4.6 Declarar una función llamada main() que recibe currentTime 3](#_Toc96079654)

[4.7 Declarar requestAnimationFrame para iniciar el primer render 4](#_Toc96079655)

[4.8 Crear la función update() 4](#_Toc96079656)

[4.9 Crear la función draw() 4](#_Toc96079657)

[4.10 Crear la función checkDeath() 4](#_Toc96079658)

[5 food.js 5](#_Toc96079659)

[5.1 Agregar import de onSnake y expandSnake 5](#_Toc96079660)

[5.2 Agregar import de randomGridPosition 5](#_Toc96079661)

[5.3 Declarar variable let de food 5](#_Toc96079662)

[5.4 Declarar constante llamada EXPANSION\_RATE 5](#_Toc96079663)

[5.5 Crear una función de tipo export llamada update() 5](#_Toc96079664)

[5.6 Crear una función de tipo export llamada draw() que recibe gameBoard 5](#_Toc96079665)

[5.7 Crear una función llamada getRandomFoodPosistion () 5](#_Toc96079666)

[6 snake.js 5](#_Toc96079667)

[6.1 Importar getInputDirection from input.js 5](#_Toc96079668)

[6.2 Crear una constante llamada SNAKE\_SPEED y agregar export 6](#_Toc96079669)

[6.3 Crear otra constante que se llama snakeBody 6](#_Toc96079670)

[6.4 Declarar una variable let llamada newSegments, para ir agregando cuantas piezas al tocar la comida 6](#_Toc96079671)

[6.5 Crear una función llamada update() y de tipo export 6](#_Toc96079672)

[6.6 Crear una nueva función de tipo export llamada draw(), pasarle el parámetro gameBoard 6](#_Toc96079673)

[6.7 Crear una nueva función de tipo export llamada expandSnake() que recibe el parámetro amount 7](#_Toc96079674)

[6.8 Crear una nueva función de tipo export llamada onSnake() que recibe el parámetro position y declarar el valor de ignoreHead a false 7](#_Toc96079675)

[6.9 Crear una nueva función llamada equalPosition() 7](#_Toc96079676)

[6.10 Crear una nueva función llamada equalPosition() 7](#_Toc96079677)

[6.11 Crear una nueva función llamada addSegments() 7](#_Toc96079678)

[7 Input.js 8](#_Toc96079679)

[7.1 Declarar variable let llamada inputDirection 8](#_Toc96079680)

[7.2 Declarar variable let llamada lastInputDirection 8](#_Toc96079681)

[7.3 Declarar window.addEventListener 8](#_Toc96079682)

[7.4 Declarar función de tipo export llamada getInputDirection() 8](#_Toc96079683)

[8 grid.js 8](#_Toc96079684)

[8.1 Declarar constante llamada GRID\_SIZE 8](#_Toc96079685)

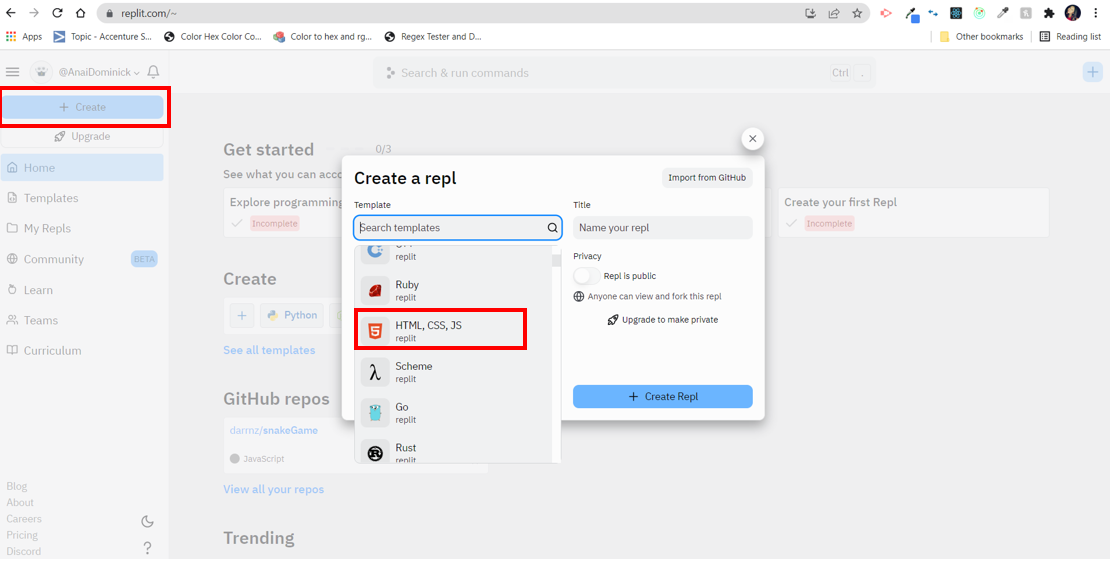
[8.2 Crear función de tipo export llamada randomGridPosition() 9](#_Toc96079686)

[8.3 Crear función de tipo export llamada outsideGrid() 9](#_Toc96079687)

# Environment Setup

## Ingresa a tu cuenta de stackblitz

Entra a <https://replit.com/> y haz login con tus credenciales. Después en la esquina superior izquierda da click en créate y selecciona la opción HTML/CSS/JS.



# index.html

## Pre-requisites

* Borrar archivo me.html
* Renombrar archivo script.js por game.js
* Descomentar línea 5 dentro de index.html
* Borrar línea 9 dentro de archivo index.html
* Reemplazar nombre de archivo script.js por game.js dentro de archivo index.html

# style.css

## Style.css

Dentro de este archive copia los siguientes estilos:

body {

height: 100vh;

width: 100vw;

display: flex;

justify-content: center;

align-items: center;

margin: 0;

background-color: black;

}

#game-board {

background-color: #ccc;

width: 100vmin;

height: 100vmin;

display: grid;

grid-template-rows: repeat(21, 1fr);

grid-template-columns: repeat(21, 1fr);

}

.snake {

background-color:hsl(200, 100%, 50%);

border: .25vmin solid black;

}

.food {

background-color:hsl(50, 100%, 50%);

border: .25vmin solid black;

}

# game.js

## Import de SNAKE\_SPEED, update as updateSnake, draw as drawSnake

import { SNAKE\_SPEED, update as updateSnake, draw as drawSnake, getSnakeHead, snakeIntersection } from './snake.js';

## Import outsideGrid

import { outsideGrid } from './grid.js'

## Import update as updateFood, draw as drawFood

import { update as updateFood, draw as drawFood } from './food.js';

## Declara una variable let llamada lastRenderTime y gameOver

let lastRenderTime = 0;

let gameOver = false;

## Declara una constante llamada gameBoard

const gameBoard = document.getElementById('game-board');

## Declarar una función llamada main() que recibe currentTime

function main(currentTime) {

**// Reset the game**

if (gameOver) {

if(confirm('You lost. Press ok to restart')) {

window.location = '/'

}

return;

}

**// Para obtener el siguiente frame**

window.requestAnimationFrame(main);

**// Declarar una constante llamada secondsSinceLastRender**

const secondsSinceLastRender = (currentTime - lastRenderTime)/1000;

**// Abrir un if donde si es menor que 1 igualará lastRenderTime a currentTime**

if (secondsSinceLastRender < 1 /SNAKE\_SPEED) return

lastRenderTime = currentTime;

**// Llamar los métodos update y draw, es decir, actualiza y dibuja**

update()

draw()

}

## Declarar requestAnimationFrame para iniciar el primer render

window.requestAnimationFrame(main);

## Crear la función update()

function update() {

updateSnake();

updateFood();

checkDeath();

}

## Crear la función draw()

function draw() {

gameBoard.innerHTML = '';

drawSnake(gameBoard);

drawFood(gameBoard);

}

## Crear la función checkDeath()

function checkDeath() {

gameOver = outsideGrid(getSnakeHead()) || snakeIntersection()

}

# food.js

## Agregar import de onSnake y expandSnake

import { onSnake, expandSnake } from './snake.js';

## Agregar import de randomGridPosition

import { randomGridPosition } from './grid.js';

## Declarar variable let de food

let food = getRandomFoodPosistion();

## Declarar constante llamada EXPANSION\_RATE

const EXPANSION\_RATE = 1;

## Crear una función de tipo export llamada update()

export function update() {

if (onSnake(food)) {

expandSnake(EXPANSION\_RATE);

**// Nueva posicion de la comida**

food = getRandomFoodPosistion();

}

}

## Crear una función de tipo export llamada draw() que recibe gameBoard

export function draw(gameBoard) {

const foodElement = document.createElement('div');

foodElement.style.gridRowStart = food.y;

foodElement.style.gridColumnStart = food.x;

foodElement.classList.add('food');

gameBoard.appendChild(foodElement);

}

## Crear una función llamada getRandomFoodPosistion ()

function getRandomFoodPosistion() {

let newFoodPosition;

while (newFoodPosition == null || onSnake(newFoodPosition)) {

newFoodPosition = randomGridPosition();

}

return newFoodPosition;

}

# snake.js

## Importar getInputDirection from input.js

import { getInputDirection } from "./input.js";

## Crear una constante llamada SNAKE\_SPEED y agregar export

export const SNAKE\_SPEED = 5;

## Crear otra constante que se llama snakeBody

const snakeBody = [{ x: 10, y: 11 }];

## Declarar una variable let llamada newSegments, para ir agregando cuantas piezas al tocar la comida

let newSegments = 0;

## Crear una función llamada update() y de tipo export

export function update() {

**// Dentro mandar a llamar el método addSegments()**

addSegments();

**// Crear una constante llamada inputDirection e igualarla a la función getInputDirection()**

const inputDirection = getInputDirection();

**// Después crear un for**

for (let i = snakeBody.length - 2; i >= 0; i--) {

console.log(snakeBody);

snakeBody[i + 1] = { ...snakeBody[i] };

console.log(snakeBody);

}

**// Asignar los valores de los ejes x y y**

console.log(snakeBody[0]);

snakeBody[0].x += inputDirection.x;

snakeBody[0].y += inputDirection.y;

}

## Crear una nueva función de tipo export llamada draw(), pasarle el parámetro gameBoard

El parámetro de gameboard se declara dentro del archivo game.js

export function draw(gameBoard) {

**//Hacer un forEach a snakeBody**

snakeBody.forEach((segment) => {

const snakeElement = document.createElement("div");

**// Asignar el valor de las columnas y filas para dibujar serpiente**

snakeElement.style.gridRowStart = segment.y;

snakeElement.style.gridColumnStart = segment.x;

snakeElement.classList.add("snake");

gameBoard.appendChild(snakeElement);

});

}

## Crear una nueva función de tipo export llamada expandSnake() que recibe el parámetro amount

export function expandSnake(amount) {

newSegments += amount;

}

## Crear una nueva función de tipo export llamada onSnake() que recibe el parámetro position y declarar el valor de ignoreHead a false

export function onSnake(position, { ignoreHead = false } = {}) {

**// Regresa equal positions**

return snakeBody.some((segment, index) => {

if (ignoreHead && index === 0) return false;

return equalPositions(segment, position)

})

}

## Crear una nueva función llamada equalPosition()

function equalPositions(pos1, pos2) {

return pos1.x === pos2.x && pos1.y === pos2.y;

}

## Crear una nueva función llamada equalPosition()

function addSegments() {

for (let i = 0; i < newSegments; i++) {

snakeBody.push({ ...snakeBody[snakeBody.length - 1] });

}

//24.1

newSegments = 0

};

## Crear una nueva función llamada addSegments()

function addSegments() {

for (let i = 0; i < newSegments; i++) {

snakeBody.push({ ...snakeBody[snakeBody.length - 1] });

}

//24.1

newSegments = 0

};

# Input.js

## Declarar variable let llamada inputDirection

let inputDirection = { x: 0, y: 0 };

## Declarar variable let llamada lastInputDirection

let lastInputDirection = { x: 0, y: 0 };

## Declarar window.addEventListener

window.addEventListener('keydown', e => {

switch (e.key) {

case 'ArrowUp':

if (lastInputDirection.y !== 0) break; //19

inputDirection = { x: 0, y: -1 };

break;

case 'ArrowDown':

if (lastInputDirection.y !== 0) break; //19

inputDirection = { x: 0, y: 1 };

break;

case 'ArrowRight':

if (lastInputDirection.x !== 0) break; //19

inputDirection = { x: 1, y: 0 };

break;

case 'ArrowLeft':

if (lastInputDirection.x !== 0) break; //19

inputDirection = { x: -1, y: 0 };

break;

}

})

## Declarar función de tipo export llamada getInputDirection()

export function getInputDirection() {

lastInputDirection = inputDirection

return inputDirection

}

# grid.js

## Declarar constante llamada GRID\_SIZE

const GRID\_SIZE = 21

## Crear función de tipo export llamada randomGridPosition()

export function randomGridPosition() {

return {

x: Math.floor(Math.random() \* GRID\_SIZE) + 1,

y: Math.floor(Math.random() \* GRID\_SIZE) + 1

}

}

## Crear función de tipo export llamada outsideGrid()

export function outsideGrid(position) {

return (

position.x < 1 || position.x > GRID\_SIZE ||

position.y < 1 || position.y > GRID\_SIZE

)

}

# Registro segunda sesión

Completa el siguiente formulario de Google Forms para guardar tu información de contacto y mandarte la invitación de la segunda sesión.

<https://forms.gle/UzKaPaZWTv4PyfuQ6>