Ejercicios sobre JSON, Fetch y Promesas en JavaScript

Descarga y descomprime el fichero .zip con la estructura de archivos para los ejercicios de este tema.

1. Abre una consola JavaScript y carga la clase Matrix de math.js.

   a) Instancia la clase Matrix y usa su método set para crear la siguiente matriz:

```
console.log( matriz );

Matrix {
  grid:
   [ [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ],
     [ 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 ],
     [ 20, 21, 22, 23, 24, 25, 26, 27, 28, 29 ],
     [ 30, 31, 32, 33, 34, 35, 36, 37, 38, 39 ],
     [ 40, 41, 42, 43, 44, 45, 46, 47, 48, 49 ],
     [ 50, 51, 52, 53, 54, 55, 56, 57, 58, 59 ],
     [ 60, 61, 62, 63, 64, 65, 66, 67, 68, 69 ],
     [ 70, 71, 72, 73, 74, 75, 76, 77, 78, 79 ],
     [ 80, 81, 82, 83, 84, 85, 86, 87, 88, 89 ],
     [ 90, 91, 92, 93, 94, 95, 96, 97, 98, 99 ] ] }
```

Nota: intenta calcular los valores de la matriz usando los índices con los que recorres la matriz en lugar de usar una variable auxiliar.

   b) Usa el método forEach del objeto matriz para que muestre por pantalla el valor de cada celda en el formato (valor,x,y).

   Es decir, que muestre por pantalla:

```
0 0 0
1 0 1
2 0 2
3 0 3
4 0 4
5 0 5
6 0 6
7 0 7
8 0 8
9 0 9
10 1 0
11 1 1
12 1 2
13 1 3
14 1 4
15 1 5
16 1 6
17 1 7
18 1 8
19 1 9
20 2 0
...
   etc.
...
```

2- Implementa la función loadJSON que hace uso del API fetch para cargar una url y devolver una promesa que extrae el cuerpo JSON de la respuesta. (¡La solución está en los vídeos!)

```
function loadJSON(url){

}
```

3- Implementa la función loadImage que devuelve una promesa para la carga de una imagen de forma asíncrona. (¡La solución está en los vídeos!)

```
export function loadImage(url){

}
```

4- Estudia el método loadLevel() y basándote en él, implementa el método loadSpriteSheet().

```
function loadSpriteSheet(){
   // return
   //   cargar fichero JSON desde sprites.json y dejarlo en sheetSpec
   // usar Pomise.all para ejecutar las promesas sheetSpec y loadImage
   // fíjate que loadImage toma como parámetro la ruta al fichero con los sprites (esa ruta está en sheetSpec)
   .then(([sheetSpec, image]) => {
       const sprites = new SpriteSheet( // fíjate en la clase SpriteSheet y pásale los parámetros adecuados
                                   );

         // por cada tile de sheetSpec
            sprites.defineTile(
               // nombre del tile
               // posición X del tile
               // posición Y del tile
               );
         });

       return sprites;
   });
 }
```

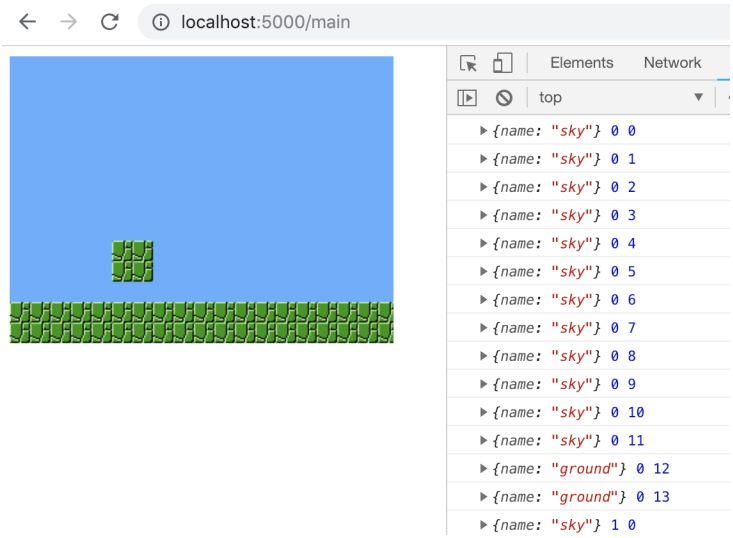
5- Estudia el cometido de la función createTiles en los vídeos sobre los ejercicios de esta semana e implementa dicha función:

```
   function createTiles(level, backgrounds){

   // para cada background
      // para cada rango
            // calcular todos los tiles que hay meter en la matriz level.tiles
               level.tiles.set(x, y, {
                  name: background.tile,
               });
      // cerrar llaves y paréntesis de forma adecuada
```

6- Añade en *main.js*, dentro del then de loadLevel, la instrucción necesaria para mostrar el contenido del mapa en consola (recuerda el ejercicio 1.b)

(Hasta que no hagas los ejercicios relacionados con el tema de Canvas, semana 6, no verás el nivel pintado en pantalla, pero en la consola sí que deberías ver el mismo resultado)



<u>Ejercicios para el tema Canvas:</u>

Canvas 1) Cambia el tamaño del canvas a 600x600. Modifica level.js para que los elementos se pinten escalados al nuevo tamaño.

Canvas 2. Modifica sprites.json para hacer uso de esta spritesheet: https://www.dropbox.com/s/f332eiuaj4uxzpb/pang.gif?dl=1 (los tiles son de 8x8 y los dos que buscamos están en [0,0] y [0,1]). Sin tocar nada más del código, comprueba que se carga esta pantalla en el canvas.