PRÁCTICA - Juegos con funciones/array/objetos en JS

Jaime García Pedrote

Manual de juego

¿Has pensado alguna vez en desquitarte con un ser vivo sin consecuencias?

¿Reventar un animal a mazazos sin ser perseguido por PETA? (espero que no)

Pues dejame decirte que este es tu juego, en matatopos podrás cumplir tu más depravada afición reprimida.

El objetivo a cumplir es reventar tantísimos topos como te sea posible hasta alcanzar la soñada meta de los **1000 puntos** evitando golpear a los inocentes perros oficinistas que disminuirán tu puntuación (hasta por debajo de 0 así que cuidado).

Para comenzar el juego debes pulsar el botón de “Nuevo Juego” y desatar el caos. A más veces lo pulses **MÁS** bichillos tendrás por pantalla al mismo tiempo. ¿Podrás tener la situación bajo control?

Manual Técnico

El juego entero es un objeto llamado “Matatopos” con dos atributos principales, un método principal y diversos métodos auxiliares.

Estos atributos son un array para saber qué casillas se encuentran libres en la tabla donde los topos son generados y la puntuación que es mostrada en pantalla.

```
ocupacion: [], //Para llevar las posiciones
puntuacion: 0, //Puntuación inicial
```

Antes de entrar en materia y hablar sobre los métodos implementados, es importante ver la estructura principal del html:

```
<body>
  <h1 id="score">SCORE: 0</h1>
  <table>
    <tr>
      <td id="1"></td>
      <td id="2"></td>
      <td id="3"></td>
      <td id="4"></td>
      <td id="5"></td>
      <td id="6"></td>
      <td id="7"></td>
      <td id="8"></td>
      <td id="9"></td>
      <td id="10"></td>
    </tr>
    <tr>
      <td id="11"></td>
```

```
      <td id="49"></td>
      <td id="50"></td>
    </tr>
  </table>

  <button
    onclick="setInterval(() => {
      matatopos.lanzaImagen()
  }, 350);"
  >
    Nuevo Juego
  </button>
</body>
```

Tenemos un título que muestra la puntuación, la tabla con un id para cada casilla y el botón que ejecuta el método “lanzaImagen” (ya lo veremos a continuación) cada cierto periodo de tiempo.

En lo que a los métodos respecta, comencemos con los auxiliares:

```
casilla: function () {
    //Genera un número aleatorio entre 1 y 50
    return Math.floor(Math.random() * 50 + 1);
},
```

Este método genera un número aleatorio entre el 1 y el 50, lo usaremos para sacar el id de las casillas de la tabla y que las imágenes salgan aleatoriamente.

```
huyeImagen: function () {
    //Elimina la imagen
    let lugar = document.getElementById(this.ocupacion.shift());
    lugar.removeChild(lugar.firstChild);
},
```

Este método saca el id que más tiempo lleva almacenado en el array “ocupación”, dejando esa casilla libre dentro del array y además borra los elementos hijos de dicha casilla que en este caso sería una imagen.

```
sumaPuntos: function () {
    //Suma puntos
    this.puntuacion += 25;
},

restaPuntos: function () {
    //Resta puntos
    this.puntuacion -= 100;
},
```

Estos métodos alteran la puntuación total, son llamados dependiendo de la imagen sobre la que clickes pero lo veremos más adelante.

```
actualizaScore: function () {
  //Actualiza la puntuación en pantalla y enseña el mensaje de victoria
  let score = document.getElementById("score");
  if (this.puntuacion >= 1000) {
    score.innerHTML = `¡LLEGASTE A 1000 PUNTOS, IMPRESIONANTE!`;
    setTimeout(() => location.reload(), 3000);
  } else {
    score.innerHTML = `SCORE: ${this.puntuacion}`;
  }
},
```

Y por último el método que finaliza el juego, actualiza la puntuación en el “h1” que vimos en el HTML hasta que llegas a los 1000 puntos, en este caso cambia el mensaje por uno de victoria y pasados 3 segundos recarga la página para poder jugar de nuevo.

```
lanzaImagen: function () {
  //Lanza imágenes de forma aleatoria evitando repetir casillas
  let img = document.createElement("img");
  if (55 >= Math.floor(Math.random() * 100 + 1)) {
    //Decide la imagen aleatoriamente con un 55% de topo
    img.src = "/img/Rese_T._Ado.png";
    img.addEventListener("click", () => this.sumaPuntos());
  } else {
    img.src = "/img/Isabelle.png";
    img.addEventListener("click", () => this.restaPuntos());
  }
  img.addEventListener("click", () => img.remove());
  img.width = "150";

  let lugar = this.casilla();
  let place = document.getElementById(lugar);
  if (this.ocupacion.indexOf(lugar) == -1) {
    let self = this;
    setTimeout(() => this.huyeImagen(), 1500); //Elimina la imagen pasado un tiempo asegurando que exista
    this.ocupacion.push(lugar);
    place.appendChild(img);
  }
  this.actualizaScore();
},
```

Empieza la fiesta.
Para una mejor comprensión de aquí nuestro método principal voy a dividir la explicación por partes.

```
let img = document.createElement("img");
if (55 >= Math.floor(Math.random() * 100 + 1)) {
  //Decide la imagen aleatoriamente con un 55% de topo
  img.src = "/img/Rese_T._Ado.png";
  img.addEventListener("click", () => this.sumaPuntos());
} else {
  img.src = "/img/Isabelle.png";
  img.addEventListener("click", () => this.restaPuntos());
}
img.addEventListener("click", () => img.remove());
img.width = "150";
```

En esta primera parte creamos una variable local llamada “img”, que no es más que un nuevo elemento de imagen creado. A continuación generamos un número entre el 1 y el 100 para simular un porcentaje, si este es menor de 55 la imagen será la de un topo, en caso contrario un perro, simulando así un 55% de probabilidad de que aparezcan topos.

En función de la imagen resultante, se crea un evento de escucha en click sobre esta llamando al método correspondiente para sumar o restar puntos. Ambas imágenes independientemente del resultado deben tener un tamaño de 150 para encajar correctamente en las casillas y un evento de escucha en click para eliminar la imagen.

```
  let lugar = this.casilla();
  let place = document.getElementById(lugar);
  if (this.ocupacion.indexOf(lugar) == -1) {
    let self = this;
    setTimeout(() => this.huyeImagen(), 1500); //Elimina la imagen pasado un tiempo asegurando que exista
    this.ocupacion.push(lugar);
    place.appendChild(img);
  }
  this.actualizaScore();
},
```

En la segunda mitad, primero asignamos a una variable local “lugar” un número aleatorio con el método “casilla”, buscamos la casilla con el id guardado en “lugar” y lo almacenamos en “place”.

Comprobamos que el número generado y almacenado en “lugar” no se encuentre en el atributo “ocupacion”, de ser así, lo indexa al array por el final y añade el elemento hijo “img” en la casilla guardada en “place”.

Pasado un tiempo se llama al método “huyeImagen” para borrar la imagen que más tiempo tiempo lleva almacenada, tiene que ser llamado cuando se indexa otra imagen porque si se genera un número repetido no indexará imagen pero si borraría una, disminuyendo gradualmente el número de elementos en pantalla.

```css
body {
    background-image: url(/img/background.png);
    cursor: url('
}

table,th,td {
    margin: auto;
    margin-top: 50px;
}

tr,td {
    width: 150px;
    height: 150px;
    align-items: center;
}

h1 {
    text-align: center;
    font-size: 40px;
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
}
```

Finalmente añadí un poco de css para modificar la imagen del cursor a la de un martillo, dar tamaño a las celdas, añadir un fondo y resaltar la puntuación,