MEMORIA PAC DE DESARROLLO

CFGS Desarrollo de Aplicaciones Web

**Carrera de Coches**

**Autor:** Carlos Francisco Caruncho Serrano

**Tutor:** Mario Gago

**Fecha de entrega:** 20/04/2023



Se ha dividido la práctica en 3 archivos principales:

* **index.html** 🡪 contiene la estructura de la página
* **script.js** 🡪 contiene la funcionalidad
* **styles.css** 🡪 contiene la hoja de estilos para el diseño

La estructura de carpetas es:

..\pac-carreracoches 🡪 directorio raíz

..\pac-carreracoches\img 🡪 directorio de imágenes

..\pac-carreracoches\css 🡪 directorio de hojas de estilo

..\pac-carreracoches\js 🡪 directorio de código javascript

..\pac-carreracoches\sounds 🡪 directorio de sonidos

**index.html**

En la cabecera(head) colocamos el título y el icono en el navegador.

  <!--Título de la página-->

  <title>Pixar Cars Race</title>

  <!--Icono de la página-->

  <link rel="icon" type="image/png" href="/img/favicon.png" />

Cargamos las hojas de estilo CSS que se aplicarán para el diseño de la interfaz web. Una de ellas es local en el archivo “styles.css” y la otra es remota, para cargar la fuente de letra de Google “Audiowide” que usaremos globalmente en la aplicación.

 <!--Carga las hojas de estilos asociada para el diseño-->

  <link rel="stylesheet" href="css/styles.css" />

  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Audiowide|Sofia|Trirong">

En la sección de metadatos incluimos toda la información interna de la aplicación web. El “**viewport**” es la parte visible para el usuario de la aplicación web en cada dispositivo, e indica como debe adaptarse a la pantalla. El “**charset**” es el conjunto de carácteres para la página. Además, he incluido mi nombre cómo autor.

  <!--Sección de metadatos-->

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <meta charset="utf-8" />

  <meta name="author" content="Carlos Caruncho">

Para finalizar la cabecera he incluido los archivos javascript. Se cargarán al final de la carga HTML al contener la función document.ready dentro de los archivos de código, si no estuviese de esa manera se colocarían al final del “**body**”.

  <!--Código JavaScript incluido-->

  <script type="text/javascript" src="js/jquery.js"></script>

  <script type="text/javascript" src="js/script.js"></script>

</head>

**Jquery** es una librería javascript que nos facilita la escritura de código al interactuar con el DOM o estructura HTML de la página.

El segundo archivo es dónde escribo la funcionalidad principal de la aplicación.

En el cuerpo tenemos una primera sección **“div”** llamada **“navigation-bar**” que contiene el logotipo principal y las imágenes decorativas de “**Cars**”. Las imágenes se han convertido al formato **“webp”** con lo que se ha reducido su tamaño significativamente para optimizar la carga de la web.

<body>

  <!--Barra de navegación y logo-->

  <div id="navigation-bar">

    <img id="car-logo1" src="../img/rayomacqueen.webp" alt="Rayo MacQueen" class="car-logo">

    <img id="main-logo" src="../img/logo.webp" alt="Cars Logo" class="car-logo">

    <img id="car-logo2" src="../img/ramone.webp" alt="Ramone" class="car-logo">

  </div>

En la siguiente sección tenemos el formulario con la lista desplegable, dónde seleccionamos la cantidad de coches, y la botonera con los botones **“Iniciar”** y **“Reiniciar”.**

 <!--Formulario de selección de cantidad de coches que generamos-->

  <form action="" method="post" id="racing-form">

    <label>Selecciona el número de coches que participan</label>

    <select id="list-car-amount" name="list-car-amount" title="Cantidad de coches">

      <option></option>

      <option>1</option>

      <option>2</option>

      <option>3</option>

      <option>4</option>

      <option>5</option>

      <option>6</option>

      <option>7</option>

      <option>8</option>

      <option>9</option>

    </select>

    <br /> <br />

    <div class="button-bar">

      <button type="button" value="Iniciar" id="start-race" class="button">Iniciar</button>

      <button type="button" value="Reiniciar" id="reset-race" class="button">Reiniciar</button>

    </div>

  </form>

En la siguiente sección se coloca una división contenedora, para la generación de las carreteras, que podrá contener hasta 9 filas con su carretera, inicio y final de carrera.

<!--Contenedor de carreteras-->

  <div id="road-container" class="road-container"></div>

El siguiente código sirve para generar un panel modal con una tabla, donde podemos ver el orden de llegada a la meta de los coches. Dispone de un botón para cerrar el cuadro modal.

  <!--Contenedor de carreteras-->

  <div id="road-container"></div>

  <div id="modal-result-panel">

    <h2 id="classification-title">Clasificación</h2>

    <!--Tabla de resultados con-->

    <table id="classification-board">

      <tr>

        <th>Posición</th>

        <th>Participante</th>

      </tr>

    </table>

    <div id="close-container"><button type="button" value="Cerrar" id="close" class="button">Cerrar</button></div></div>

**scripts.js**

Uso la biblioteca **Jquery** para facilitar el trabajo y hacer un código más limpio y óptimo.

Con el método **“ready”** ejecutamos el código javascript cuando el DOM está totalmente cargado.

En primer lugar, inicializamos las variables globales. La cantidad de coches, el array de coches, el identificador del contenedor de carreteras, y el ancho de la ventana del explorador, el panel modal de resultados, y los botones de control.

//Carga el código con .ready cuando acaba de cargar el código html

$(document).ready(() => {

  //Inicia las variables

  let carAmount = 0; //Variable de cantidad de coches

  let cars = []; //Array de coches

  const roadContainer = $("#road-container"); //Contenedor de las carreteras

  const carAmountSelector = $("#list-car-amount"); //Lista desplegable de cantidad de coches

  const screenWidth = $(window).width(); //Recoge el ancho de la ventana del explorador

  const modalResultPanel = $("#modal-result-panel"); //Ventana modal con los resultados de la carrera

  const classificationBoard=$("#classification-board") //Cuadro de clasificación

  const btStartRace = $("#start-race"); //Botón de inicio de carrera

  const btResetRace = $("#reset-race"); //Botón de reinicio de carrera

  const btCloseModal = $("#close"); //Botón de cerrado de la ventana modal

Asociamos el evento **“change”** de la lista desplegable a una función interna. Esto significa que al cambiar el valor de la lista se ejecute esa función con una expresión lambda, le pasamos como parámetro el evento.

Al iniciarse el evento indicamos que no se recargue la página completa al cambiar la lista desplegable con el método **“preventDefault()**”.

Dentro de la función que se ejecuta el evento escondemos el botón reiniciar con el método “hide()” si fuera visible, y vaciamos si estuviera lleno el contenedor de carreteras. Esto se agrega para reiniciar la carrera una vez iniciada por primera vez.

 //Crea las carreteras y los coches según el número seleccionado

  carAmountSelector.change((event) => {

    event.preventDefault(); //Previene que se recargue la página

    //Esconde el boton resetear, y muestra el de resetear.

    btResetRace.hide();

    btStartRace.show();

    roadContainer.empty(); //Vacia el contenedor de carreteras si estuviese pintado.

A continuación, se recorre con un bucle **“for”,** que recorre una variable desde 0 a “carAmount”, que es el número de coches seleccionado en el selector desplegable.

Se crean los campos “adhoc” en el array. **“id”** con el nombre del coche. Se usa la cadena “Car”

El campo **“speed”,** usamos **Math.random()** que genera un número aleatorio entre 0 (inclusivo) y 1 (exclusivo), y luego se multiplica por 10 para obtener un número entre 0 (inclusivo) y 10 (exclusivo). Luego, **Math.floor()** redondea ese número hacia abajo al entero más cercano. Finalmente, se le suma 1 para obtener un número entre 1 y 10 (inclusivo).

Ejemplos: 4, 6, 3, 10 …

 /\* Un bucle recorre la cantidad indicada y pinta las imagenes

     \* de carreteras y los coches.

     \* Calcula la velocidad del coche con un número aleatorio del 1 al 10

     \*/

    carAmount = parseInt(carAmountSelector.val());

    for(i = 0; i <= carAmount - 1; i++){

      cars[i] = {

        id: `car` + (i + 1),

        speed: Math.floor(Math.random() \* 10 + 1),

        img: `img/car` + (i + 1) + `.png`,

      };

Creamos una variable **“newRoad”** que almacena un **“div”** de clase **“roadContainer”,** que tiene una imagen de fondo por CSS de una carretera.

    //Crea una nueva carretera y se le agrega al contenedor.

const newRoad = $("<div>").addClass("road").appendTo(roadContainer);

A continuación, añade a la carretera una línea de salida y otra de meta, y el coche.

 //Añade la línea de salida

      $("<img>")

        .attr({

          id: "start",

          src: "img/start.png",

          class: "line",

        })

        .appendTo(newRoad);

      //Añade la línea de meta a la carretera

      $("<img>")

        .attr({

          id: "finish",

          src: "img/finish.png",

          class: "line",

        })

        .appendTo(newRoad);

        //Añade el coche a la carretera

      $("<img>")

        .attr({

          id: cars[i].id,

          src: cars[i].img,

          class: "car",

        })

        .appendTo(newRoad);

Al darle con el botón derecho del ratón al botón **“startRace”,** se inicializa la variable **“finishedCar”**. Recorremos el array de coches con un “**forEach”**. Almacenamos en otra variable llamada “**carElement**” el puntero a la imagen de coche que corresponde con el elemento del bucle, y lo animamos con el método **“animate**()”. Esta función tiene como primer parámetro el margen izquierdo 0, que hemos inicializado en el css, y le decimos que lo modifique hasta el ancho de la ventana del navegador, restándole el largo del coche. El segundo parámetro es el tiempo que tarda en realizarse la animación, que surge del campo **speed** almacenado en el array de coches. El problema es que el tiempo es en milisegundos, y nuestros valores son desde 1 a 10, por lo que tenemos que multiplicarlo por 500 ms, para obtener valores óptimos para que la animación se vea más despacio. Los métodos **fadeIn** y **fadeOut**, hacen una transición de salida o entrada.

/\* Al presionar el botón iniciar carrera asigna la cantidad de coches

   \* a una variable. Luego recorre la cantidad con un bucle y anima los coches

   \* para recorrer las carreteras.

   \*/

  btStartRace.click(() => {

    let finishedCar = 0;

    /\*Recoge la cantidad de coches de la lista seleccionable y recorre

     \* un bucle con esa cantidad animando las imagenes de coches con

     \* a una velocidad aleatoria en milisegundos hasta el borde derecho.

     \*/

    cars.forEach((car) => {

      // Crea el nombre de coche en cada vuelta del bucle y lo almacena

      const carElement = $("#" + car.id);

      carElement.animate(

        {

          "margin-left": screenWidth - carElement.width() - 25, //Aumenta la distancia del margen izquierdo del coche con el tiempo

        },

        {

          duration: car.speed \* 500, //Tiempo en ms

          /\*Al completarse llama a una función callback, que suma +1 a los coches

           \* finalizados, si la variable coches finalizados es igual a la cantidad

          \* de coches seleccionados, se muestra el botón de reset y el botón reiniciar

           \*/

          complete: () => {

            finishedCar += 1;

            if (finishedCar == carAmount) {

              showResult();

              btResetRace.fadeIn();

            }

          },

        }

      );

    });

Lo siguiente es agregarle un sonido de coche de carreras de fondo. Almacenamos en una variable la ruta al sonido, en este caso mp3. Finalmente ejecutamos el sonido con el método **play().**

//Crea una variable con un sonido almacenado y lo ejecuta cuando

    //presionamos el botón iniciar

    let sound = new Audio("sounds/racecarsound.mp3");

    sound.play();

    //Esconde y muestra los botones

    btStartRace.fadeOut();

  });