MEMORIA PAC DE DESARROLLO

CFGS Desarrollo de Aplicaciones Web

Carrera de Coches

Autor: Carlos Francisco Caruncho Serrano

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>
```

```
    </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()”.

```
/Crea las carreteras y los coches según el número seleccionado
  carAmountSelector.change((event) => {
    event.preventDefault(); //Previene que se recargue la página
```

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

```
    //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.

```
  cars = []
  carAmount = parseInt(carAmountSelector.val());
```

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 imágenes
   * de carreteras y los coches.
   * Calcula la velocidad del coche con un número aleatorio del 1 al 10

  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;
   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()” de la clase Audio.

```
//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();
});
```

Creamos la función “showResult” que mostrará finalmente el panel de clasificación. En primer lugar, se reordena el array de mayor a menor en función de la velocidad, ya que

Por lógica el coche más veloz llegará antes y se mostrarán las posiciones con la imagen de los coches al lado.

```
function showResult() {
  cars.sort((a, b) => a.speed - b.speed);
```

A continuación, recorremos el array pasándole como parámetros con expresión lambda el elemento coche actual y el índice. Por cada elemento crea una fila de la tabla, y por cada fila un elemento img de la clase car-panel-item con los atributos src e id de carItem.

```
  cars.forEach((carItem, index) => {
    const row = $("<tr>");
    const position = $("<td>").text(index + 1);
    const carImage = $("<img>").addClass("car-panel-item");
    carImage.attr({
      src: carItem.img,
      id: carItem.id,
    });
```

Para finalizar la función showResult, agregamos por cada fila creada la posición y la imagen de cada coche, y cada fila a la tabla de clasificaciones, y mostramos la ventana modal de resultados con una transición de entrada.

```
//Se agrega a la fila el elemento y la imagen
    row.append(position, carImage);
    //Se agrega la fila al tablón de la clasificación
    classificationBoard.append(row);
```

```
  });

  // Muestra el panel modal en la pantalla
  modalResultPanel.fadeIn();
}
```

Ahora vamos a controlar el evento click del botón Reiniciar del formulario, el cual reiniciará los coches a la posición de salida. Asociamos el evento a la función anónima.

En primer lugar, creamos un bucle para recorrer el array cars pasándole solo el elemento car. Por cada coche, creamos una variable donde almacenar la referencia a cada imagen de coche generada, a continuación, las animamos moviéndolas hacia el lado izquierdo, reduciendo el margen izquierdo hasta 0. La duración será la misma para todos los coches.

```
 btResetRace.click(() => {
    cars.forEach((element) => {
      // Crea el nombre de coche en cada vuelta del bucle y lo
almacena
      const carElement = $(`#${element.id}`);
      carElement.animate(
        {

        "margin-left": 0, //Aumenta la distancia del margen izquierdo
del coche con el tiempo
        },
        {
          duration: 1000, ///Tiempo en ms en completar la animación
        }
      );
    });
```

Para finalizar, capturamos el evento click del reiniciar vaciamos la tabla de clasificación, hacemos desaparecer el botón

```
// Ocultar ventana modal al hacer clic en el botón de cerrar
 btCloseModal.click(() => {
   //Oculta la ventana modal
```

```
    modalResultPanel.fadeOut();
    // Borra el contenido de la tabla de clasificaciones
    classificationBoard.empty();
    btResetRace.show(); //Muestra el botón de reiniciar.
  });
});
```

styles.css

Todos los elementos de la página utilizan la fuente "Audiowide", blanca y el tamaño de fuente se establece en 20px. El fondo de la página está establecido en un color verde azulado (#b9cec4). El tamaño del fondo es de 100vw (medida porcentual del ancho gráfico del navegador)

```
* {
  font-family: "Audiowide", sans-serif;
}

body {
  font-size: 20px;
  background-color: #b9cec4;
  background-size: 100vw;
  color: white;
  padding: 0.5%;
}
```

El único h1 de la página está centrado y tiene un borde punteado de 2px de ancho y un radio de borde de 10px.

```
h1 {
  text-align: center;
  text-shadow: 10px;
  border-radius: 10px;
  border: 2px dotted white;
 }
```

Hay una imagen de carretera de fondo con una altura de 100px y una anchura del 98% del ancho de la ventana.

```css
.road {
  background-image: url(../img/road.png);
  background-size: 200px;
  height: 100px;
  width: 98vw;
}
```

Los coches se muestran con una imagen y tienen un tamaño del 90% de la altura del contenedor.

```css
.car {
  position: relative;
  margin-left: 0px;
  height: 90%;
  z-index: 3;
}
```

Los botones están ocultos con un color de fondo verde oscuro, un borde de 2px de ancho en color azul oscuro y texto blanco. Al pasar el mouse sobre el botón, el fondo cambia a blanco y el texto cambia a verde oscuro.

```css
.button {
  display: none;
  background-color: rgb(82, 146, 103);
  border: 2px darkblue wheat;
  border-radius: 10px;
  color: aliceblue;
  font-size: 18px;
  text-decoration: none;
  padding: 10px 20px;
  cursor: pointer;
}

.button:hover {
  background-color: rgb(255, 255, 255);
  color: rgb(13, 146, 91);
}
```

Hay logos de coches con un ancho del 10% y una altura del 20% del contenedor y un margen y relleno de 0.5%. Elemento logo volteado horizontalmente en 180º sobre el eje x para verlo al revés.

```
.car-logo {
  width: 10%;
  height: 20%;
  margin: 0.5%;
  padding: 0.5%;
}

#car-logo2 {
  transform: scaleX(-1);
}
```

Hay una línea vertical de salida que representa la posición inicial de los coches en la carretera, y otra al final.

```
.line {
  position: absolute;
  width: 40px;
  height: inherit;
  z-index: 1;
}
```

El formulario para que los usuarios ingresen la cantidad de coches que participarán en la carrera, tiene el color de fondo gris, borde de 5 píxeles, línea continua y verde, la alineación del texto centrada, y borde interior del 2%.

```
#navigation-bar {
  text-align: center;
  background-image: url(../img/grandstand.webp);
  background-size: 100vw;
  border: 5px solid #162c21;
  margin: 1% 0.2% 2% 0.2%;
  padding: 0.5%;
  opacity: 90%;
}
```

La barra de navegación tiene una imagen de fondo de una tribuna y un borde de 5px de ancho en color marrón oscuro.

Hay una tabla de resultados de carrera con celdas centradas y una fila de fondo gris claro cada dos filas.

Hay un panel modal que se muestra al final de la carrera con los resultados y un botón para cerrarlo.

```
#close {
 text-align: center;
 display: block;
}
```

Se pintan los coches pequeños en el tablón de clasificación. Aquí se configuran el alto y el ancho.

```
/*
 * Cada coche del panel modal de clasificación mide
 * un alto de 50 píxeles y 100 píxeles
*/
.car-panel-item {
  height: 50px;
  width: 100px;
}
```

La tabla de clasificación tiene un ancho del 100% de la ventana modal. Y el texto se alinea en el centro en toda la tabla.

```
/*Tabla de resultados de la carrera*/
table {
  width: 100%;
  text-align: center;
}
```

Cada celda de la tabla mide un 40% de ancho, y un alto de un 20%. Las líneas se alternan el color blanco con el verde, para distinguirlas mejor.

```
td {
 width: 40%;
```

```
 height: 20%;
}

tr:nth-child(even) {
  background-color: #dddddd;
  color: black;
}
```

La ventana modal está escondida, fijada en el centro para que al mover el scroll no se mueva de su posición inicial. El ancho es un 40%, y el alto es de 60%. Los márgenes interiores y exteriores son un 0,5%. El color de fondo es verde y el borde verde oscuro, sólido y de 5 pixeles de ancho. Y el alto del eje Z es 999, y es el más alto para que se vea sobre todos los demás objetos.

La distancia desde la parte superior es del 50% y de la parte izquierda es del 50% también. Para centrarlo usamos “transform”. El texto está centrado también.

```
/*Panel modal y botón para cerrarlo*/
#modal-result-panel {
  display: none;
  position: fixed;
  width: 40%;
  height: 60%;
  margin: 0.5%;
  padding: 0.5%;
  background-color: rgb(109, 124, 120);
  border: 5px solid #162c21;
  z-index: 999;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%); /* Centra la ventana modal */
  text-align: center;
}
```

Esto representa la división donde está el botón cerrar. El texto se alinea al centro con un margen 0 posición relativa con un 10% de distancia con la parte superior, y un 40% con la parte izquierda. En conjunto, -ms-transform: translateY(-50%); transform: translateY(-50%); movería el elemento verticalmente en un 50% de su propia altura, y

funcionaría en Internet Explorer 9 y versiones posteriores, así como en otros navegadores modernos.

```
#close-container {
  text-align: center;
  position: relative;
  top: 10%;
  left: 40%;
  -ms-transform: translateY(-50%);
  transform: translateY(-50%);
}
```

Bibliografía

W3schools