

ACTIVIDAD DE DESARROLLO

DISEÑO DE INTERFACES WEB

Sarai Olcina González
Abril 2021

PRESENTACIÓN

Se ha creado una aplicación web basada en una carrera de coches.

El usuario selecciona en un desplegable el número de participantes con los que desea iniciar la carrera. A continuación, hace clic en el botón “Iniciar” y tanto los coches como la línea de meta aparecen en pantalla.

Seguidamente, todos los coches presentados se desplazan hasta la línea de meta, cada uno de ellos con una velocidad diferente que se ha calculado de forma aleatoria. A su vez, el botón de “Iniciar” desaparece y es sustituido por el botón “Reiniciar”.

Una vez el último coche ha llegado a la meta, aparecen en pantalla las opciones de “Ver coche ganador” y “Ver tabla de resultados”.

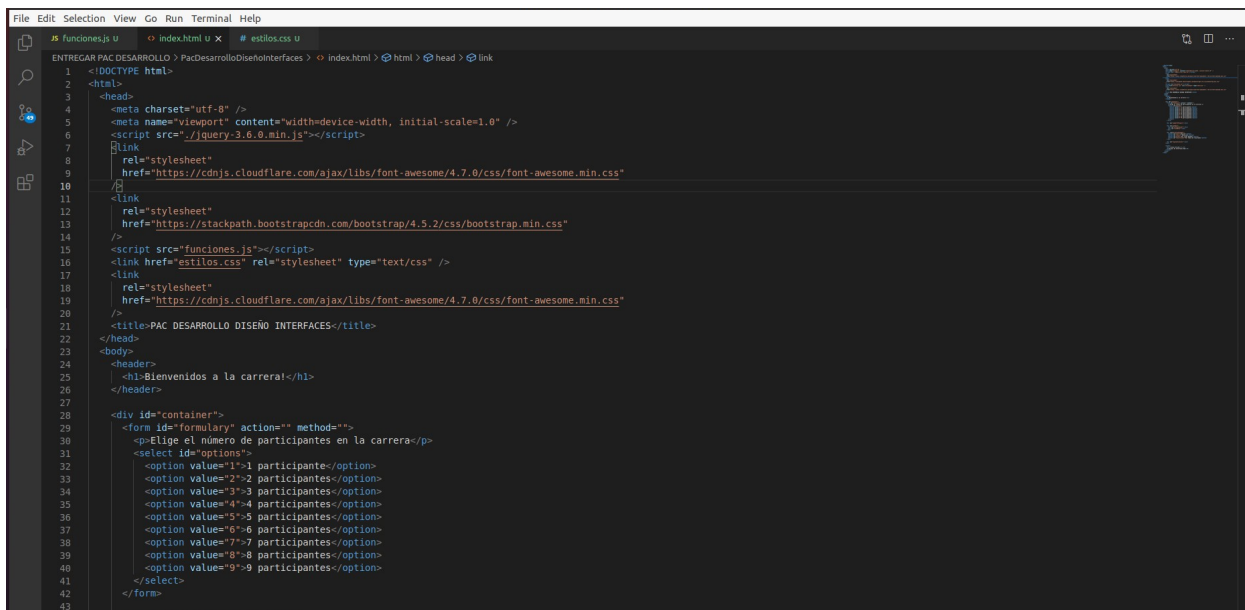
Si el usuario selecciona alguna de ellas, podrá ver cuál ha sido el coche ganador o bien una tabla con los resultados con el número de coche y velocidad de cada uno de ellos.

En caso de que se eligiera la opción “Reiniciar”, los coches seleccionados volverían al punto de partida desde donde el usuario podría nuevamente iniciar la carrera o modificar el número de coches seleccionados y por lo tanto, iniciar nuevamente la carrera pero con otro número de participantes.

COMENTARIOS DEL CÓDIGO

Se han creado 3 ficheros: index.html, funciones.js, estilos.css.
En cada uno de ellos se tratan distintos aspectos.

En primer lugar, en el archivo .html, tenemos el índice, el guión de los elementos que se van a presentar en pantalla desde el navegador.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6   <script src="//jquery-3.6.0.min.js"></script>
7   <link
8     rel="stylesheet"
9     href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"
10  />
11   <link
12     rel="stylesheet"
13     href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
14  />
15   <script src="funciones.js"></script>
16   <link href="estilos.css" rel="stylesheet" type="text/css" />
17   <link
18     rel="stylesheet"
19     href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"
20  />
21   <title>PAC DESARROLLO DISEÑO INTERFACES</title>
22 </head>
23 <body>
24   <header>
25     <h1>Bienvenidos a la carrera!</h1>
26   </header>
27
28   <div id="container">
29     <form id="formulary" action="" method="">
30       <p>Elige el número de participantes en la carrera</p>
31       <select id="options">
32         <option value="1">1 participante</option>
33         <option value="2">2 participantes</option>
34         <option value="3">3 participantes</option>
35         <option value="4">4 participantes</option>
36         <option value="5">5 participantes</option>
37         <option value="6">6 participantes</option>
38         <option value="7">7 participantes</option>
39         <option value="8">8 participantes</option>
40         <option value="9">9 participantes</option>
41       </select>
42     </form>
43   </div>
```

Figura 1: Código index.html I

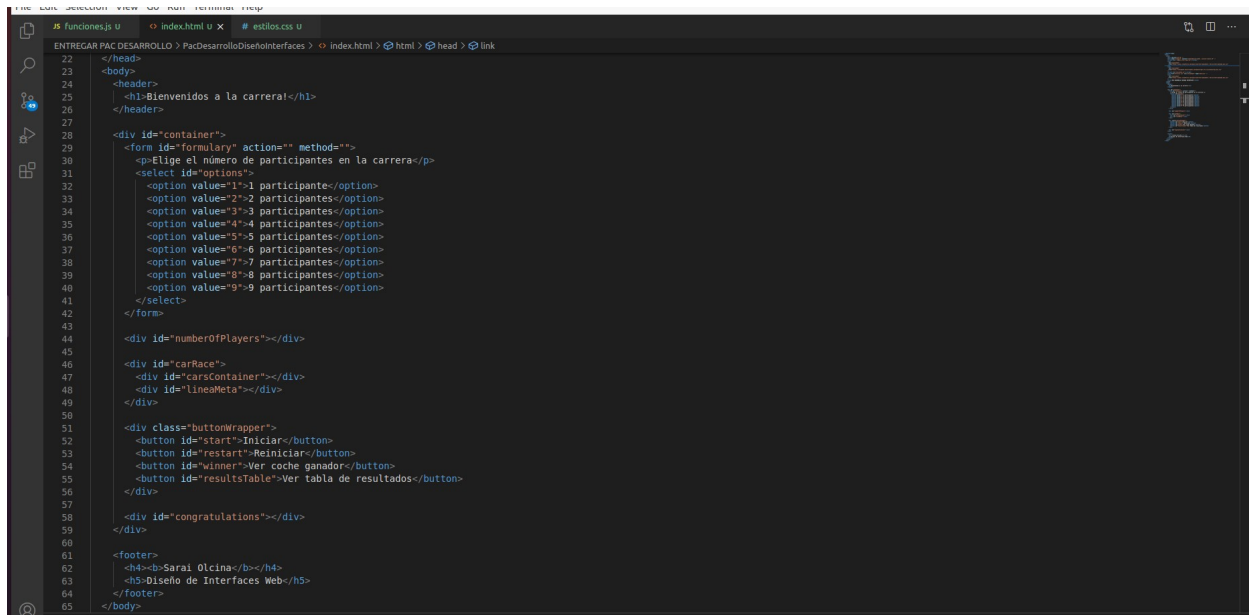
En la parte del head hemos importado las distintas librerías y archivos que vamos a utilizar, así como el título de la página.

Dentro del body, se han creado los distintos elementos con los que vamos a trabajar.

Tenemos por un lado el formulario, los botones, el footer y varios elementos div que más tarde comentaremos, pero que nos servirán para crear los coches, la línea de meta, mensajes de texto, etc.

Hemos asignado a cada elemento un identificador ("id") de forma que luego podamos "llamar" a este elemento desde el archivo js o desde el archivo de estilos y así poder incluirlo en funciones o darle un estilo determinado.

Como vemos, los contenedores de "carsContainer" y "lineaMeta" están vacíos,. Más tarde, a través de una función, los iremos rellenando una vez el usuario haya indicado el número de participantes con el que quiere jugar.



```
22 </head>
23 <body>
24 <header>
25 <h1>Bienvenidos a la carrera!</h1>
26 </header>
27
28 <div id="container">
29 <form id="formulary" action="" method="">
30 <p>Elige el número de participantes en la carrera:</p>
31 <select id="options">
32 <option value="1">1 participante</option>
33 <option value="2">2 participantes</option>
34 <option value="3">3 participantes</option>
35 <option value="4">4 participantes</option>
36 <option value="5">5 participantes</option>
37 <option value="6">6 participantes</option>
38 <option value="7">7 participantes</option>
39 <option value="8">8 participantes</option>
40 <option value="9">9 participantes</option>
41 </select>
42 </form>
43
44 <div id="numberOfPlayers"></div>
45
46 <div id="carRace">
47 <div id="carsContainer"></div>
48 <div id="lineaMeta"></div>
49 </div>
50
51 <div class="buttonWrapper">
52 <button id="start">Iniciar</button>
53 <button id="restart">Reiniciar</button>
54 <button id="winner">Ver coche ganador</button>
55 <button id="resultsTable">Ver tabla de resultados</button>
56 </div>
57
58 <div id="congratulations"></div>
59 </div>
60
61 <footer>
62 <h4>B-Saraí Olcina</h4>
63 <h5>Diseño de Interfaces Web</h5>
64 </footer>
65 </body>
```

Figura 2: Código index.html II

En el archivo funciones.js se recogen todas las funciones que vamos a utilizar. Se va a utilizar la librería JQuery (que veíamos importada en la cabecera del index.html).

Vemos que tenemos una “única” función (`$(document).ready(function(){})`; dentro de la cual están el resto de funciones.

Esta función va a empezar a ejecutarse en cuanto el documento se abra en un navegador (`.ready`).

Por eso, hemos metido todas las funciones dentro de esta única función, porque queremos tenerlas disponibles desde el momento inicial (aunque no necesariamente todas van a empezar a ejecutarse desde este momento, solo algunas de ellas lo harán).

La primera función con la que nos encontramos, es la que muestra el número de participantes una vez el usuario ha seleccionado la opción en el formulario inicial.

Cuando el usuario haga “click” en el id “options” (identificador que pertenece al formulario), se ejecutará la función.

Seleccionamos en modo texto el valor del número de participantes elegidos y lo mostramos por pantalla.

```

File Edit Selection View Go Run Terminal Help
# funciones.js u x index.html u # estilos.css u
ENTREGAR PAC DESARROLLO > PacDesarrolloDiseñoInterfaces > # funciones.js > ready() callback > on("click") callback
1 $(document).ready(function () {
2   var optionValue = 0;
3   var timeArray = [];
4   var carNameArray = [];
5   var randomTime = 0;
6
7   //Función que confirma el número de participantes seleccionados
8   $("#options").on("click", function () {
9     var optionText = $("#options option:selected").text();
10    var numberOfPlayers = document.getElementById("numberOfPlayers");
11    numberOfPlayers.innerText = "Has elegido jugar con " + optionText;
12  });
13
14  //Función que presenta el array de coches
15  $("#options").on("click", function () {
16    $("#start").show();
17    $("#formula").hide(1000);
18    $("#numberOfPlayers").show();
19    $("#lineaMeta").css("visibility", "visible");
20
21    //Obtenemos el valor de la opción seleccionada con el método val()
22    var optionValue = $("#options").val();
23    var carsContainer = document.getElementById("carsContainer");
24    while (carsContainer.hasChildNodes()) {
25      carsContainer.removeChild(carsContainer.lastChild);
26    }
27
28    for (var index = 1; index <= parseInt(optionValue); index++) {
29      var imageContainer = document.createElement("div");
30      imageContainer.id = "cars" + index;
31      carsContainer.appendChild(imageContainer);
32      var car = document.getElementById("cars" + index);
33      var img = document.createElement("img");
34      img.className = "car";
35      img.src = "img/cars" + index + ".png";
36      const lineaMeta = document.getElementById("lineaMeta");
37      if (parseInt(optionValue) > 2) {
38        img.width = "60";
39        img.height = "60";
40        img.style.padding = "6px";
41        lineaMeta.style.height = parseInt(optionValue) * 60 + "px";
42      } else {
43        img.width = "130";
44        img.height = "130";

```

Figura 3: Código funciones.js I

A continuación creamos la función que va a presentar el array de coches:

```

File Edit Selection View Go Run Terminal Help
# funciones.js u x index.html u # estilos.css u
ENTREGAR PAC DESARROLLO > PacDesarrolloDiseñoInterfaces > # funciones.js > ready() callback > on("click") callback
11    numberOfPlayers.innerText = "Has elegido jugar con " + optionText;
12  });
13
14  //Función que presenta el array de coches
15  $("#options").on("click", function () {
16    $("#formula").hide(1000);
17    $("#numberOfPlayers").show();
18    $("#lineaMeta").css("visibility", "visible");
19
20    //Obtenemos el valor de la opción seleccionada con el método val()
21    var optionValue = $("#options").val();
22    var carsContainer = document.getElementById("carsContainer");
23    while (carsContainer.hasChildNodes()) {
24      carsContainer.removeChild(carsContainer.lastChild);
25    }
26
27    for (var index = 1; index <= parseInt(optionValue); index++) {
28      var imageContainer = document.createElement("div");
29      imageContainer.id = "cars" + index;
30      carsContainer.appendChild(imageContainer);
31      var car = document.getElementById("cars" + index);
32      var img = document.createElement("img");
33      img.className = "car";
34      img.src = "img/cars" + index + ".png";
35      const lineaMeta = document.getElementById("lineaMeta");
36      if (parseInt(optionValue) > 2) {
37        img.width = "60";
38        img.height = "60";
39        img.style.padding = "6px";
40        lineaMeta.style.height = parseInt(optionValue) * 60 + "px";
41      } else {
42        img.width = "130";
43        img.height = "130";
44        lineaMeta.style.height = parseInt(optionValue) * 130 + "px";
45      }
46      car.appendChild(img);
47    }
48  });
49
50  //Función para desplazar los coches y almacenar sus valores random de velocidad
51  $("#start").click(function () {
52    optionValue = $("#options").val();
53    var counter = 0;
54

```

Figura 4: Código funciones.js II

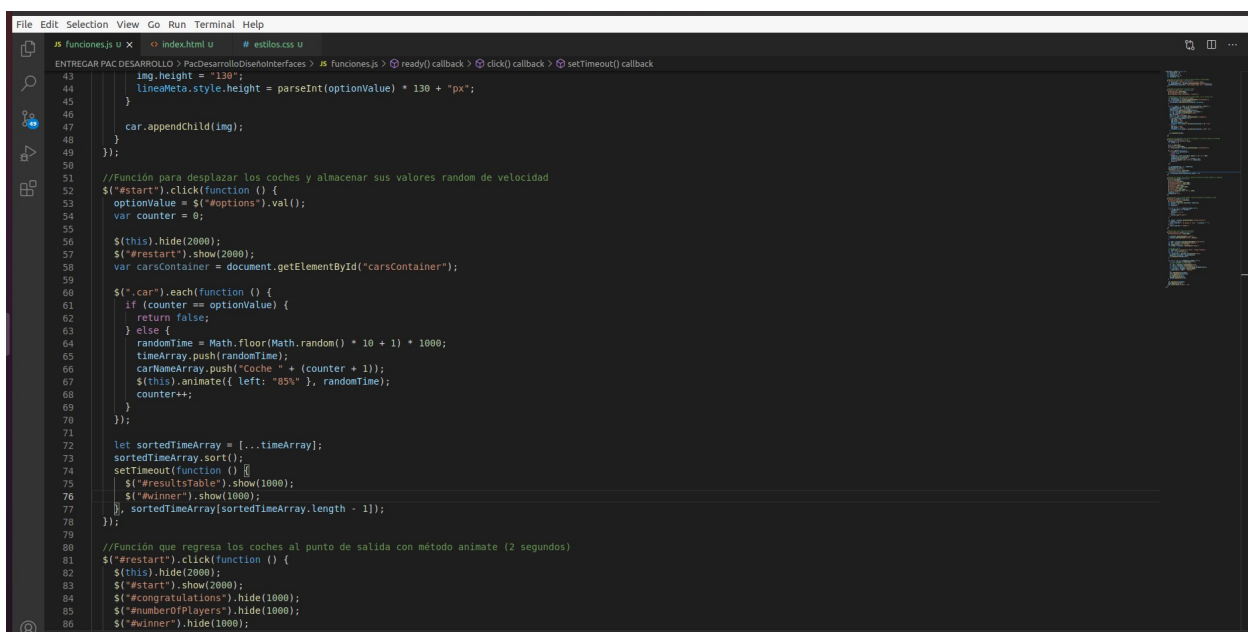
Esta función lo primero que hace es ocultar el formulario y mostrar los elementos relativos a la meta y los coches.

Una vez obtenemos el valor de la opción seleccionada por el usuario (con el método `val()`), creamos un bucle (que llega al valor elegido por el usuario). Para cada iteración de este bucle se va a crear un coche, por lo que tendremos tantos coches como participantes seleccionados.

Para cada iteración vamos a crear un elemento `div` y otro `img` y le vamos a asignar sus respectivos atributos.

En caso de que el usuario elija 2 participantes, el tamaño de los coches será distinto al resto de las opciones.

Aprovecharemos el bucle para definir el tamaño de la línea de meta, en función de si se escogen más o menos participantes.



```
File Edit Selection View Go Run Terminal Help
funciones.js u x index.html u # estilos.css u
ENTREGAR PAC DESARROLLO > PacDesarrolloDiseñoInterfaces > JS funciones.js > ready() callback > click() callback > setTimeout() callback
43     img.height = "130";
44     lineaMeta.style.height = parseInt(optionValue) * 130 + "px";
45   }
46   }
47   car.appendChild(img);
48 }
49 });
50
51 //Función para desplazar los coches y almacenar sus valores random de velocidad
52 $("#start").click(function () {
53   optionValue = $("#options").val();
54   var counter = 0;
55
56   $(this).hide(2000);
57   $("#restart").show(2000);
58   var carsContainer = document.getElementById("carsContainer");
59
60   $(".car").each(function () {
61     if (counter == optionValue) {
62       return false;
63     } else {
64       randomTime = Math.floor(Math.random() * 10 + 1) * 1000;
65       timeArray.push(randomTime);
66       carNameArray.push("Coche " + (counter + 1));
67       $(this).animate({ left: "85%" }, randomTime);
68       counter++;
69     }
70   });
71
72   let sortedTimeArray = [...timeArray];
73   sortedTimeArray.sort();
74   setTimeout(function () {
75     $("#resultTable").show(1000);
76     $("#winner").show(1000);
77   }, sortedTimeArray[sortedTimeArray.length - 1]);
78 });
79
80 //Función que regresa los coches al punto de salida con método animate (2 segundos)
81 $("#restart").click(function () {
82   $(this).hide(2000);
83   $("#start").show(2000);
84   $("#congratulations").hide(1000);
85   $("#numberOfPlayers").hide(1000);
86   $("#winner").hide(1000);
87 });
```

Figura 5: Código funciones.js III

En cuanto a la función que desplaza los coches:

En primer lugar, oculta el botón de Iniciar y muestra el de Reiniciar (que como veremos en la hoja de estilos, estaba oculto por defecto).

A continuación, a través del bucle `each` de JQuery, vamos a mover cada coche.

En primer lugar, le pasamos como parámetro la clase entera `“car”` (era uno de los atributos que habíamos creado en la función anterior para cada coche). De esta forma, esta función se va a aplicar a todos aquellos elementos que contengan la clase `“car”`.

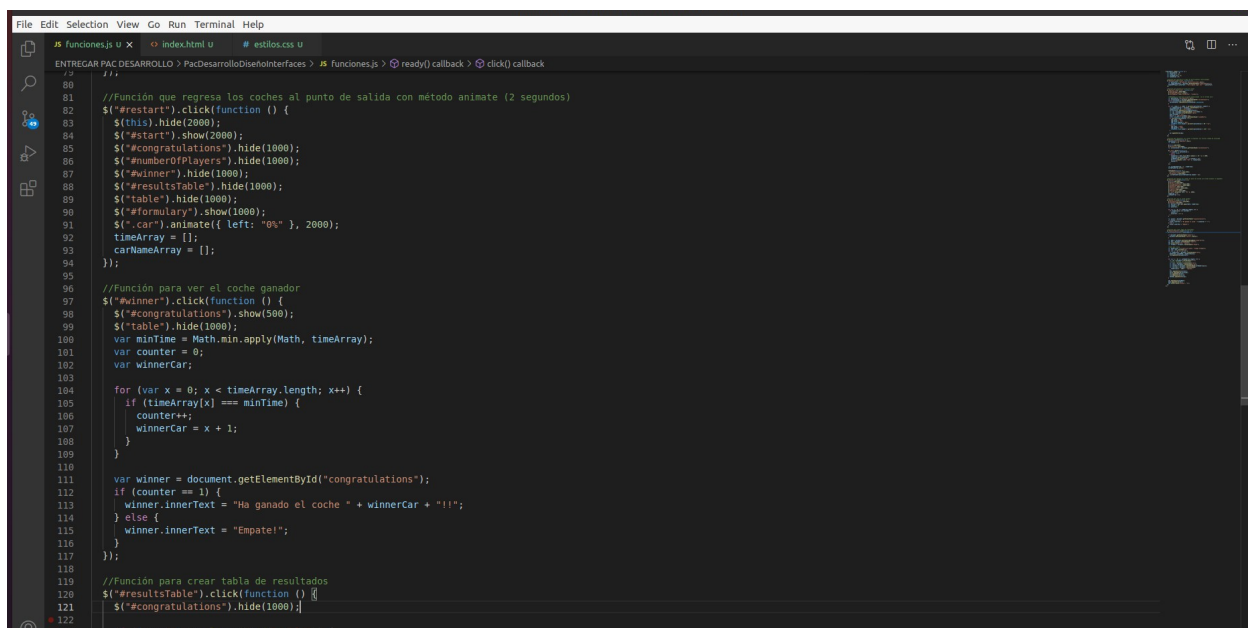
Usaremos el método `animate` para mover los coches, el cuál recibe como parámetros la posición y el tiempo. Le indicamos la misma posición que tiene la línea de meta (un par de puntos más para

que quede inmediatamente después) y en cuanto al tiempo, es un valor aleatorio entre 1 y 10 previamente calculado con el método `.random()`.

Aprovechamos y guardamos este tiempo en el array `timeArray` que más tarde utilizaremos para mostrar los resultados.

Ordenamos en otro array (`sortedTimeArray`) los tiempos, de forma que cuando llegue el último de los coches, se muestren los botones para ver el coche ganador y los resultados.

En caso de que el usuario seleccione la opciones de “Reiniciar”, usaríamos de nuevo el método `animate` para devolver los coches al punto inicial (0%) en un tiempo de 2 segundos, además de mostrar y ocultar los botones correspondientes y vaciar los arrays para que se vuelvan a rellenar con los nuevos valores que se elijan.



```
File Edit Selection View Go Run Terminal Help
# funciones.js u X index.html u # estilos.css u
ENTREGAR PAC DESARROLLO > PacDesarrolloDiseñoInterfaces > # funciones.js > ready() callback > click() callback
//
80
81 //Función que regresa los coches al punto de salida con método animate (2 segundos)
82 $("#restart").click(function () {
83   $(this).hide(2000);
84   $("#start").show(2000);
85   $("#congratulations").hide(1000);
86   $("#numberOfPlayers").hide(1000);
87   $("#winner").hide(1000);
88   $("#resultsTable").hide(1000);
89   $("#table").hide(1000);
90   $("#formulaary").show(1000);
91   $(".car").animate({ left: "0%", 2000);
92   timeArray = [];
93   carNameArray = [];
94 });
95
96 //Función para ver el coche ganador
97 $("#winner").click(function () {
98   $("#congratulations").show(500);
99   $("#table").hide(1000);
100   var minTime = Math.min.apply(Math, timeArray);
101   var counter = 0;
102   var winnerCar;
103
104   for (var x = 0; x < timeArray.length; x++) {
105     if (timeArray[x] === minTime) {
106       counter++;
107       winnerCar = x + 1;
108     }
109   }
110
111   var winner = document.getElementById("congratulations");
112   if (counter == 1) {
113     winner.innerHTML = "Ha ganado el coche " + winnerCar + "!";
114   } else {
115     winner.innerHTML = "Empate!";
116   }
117 });
118
119 //Función para crear tabla de resultados
120 $("#resultsTable").click(function () {
121   $("#congratulations").hide(1000);
122 }
```

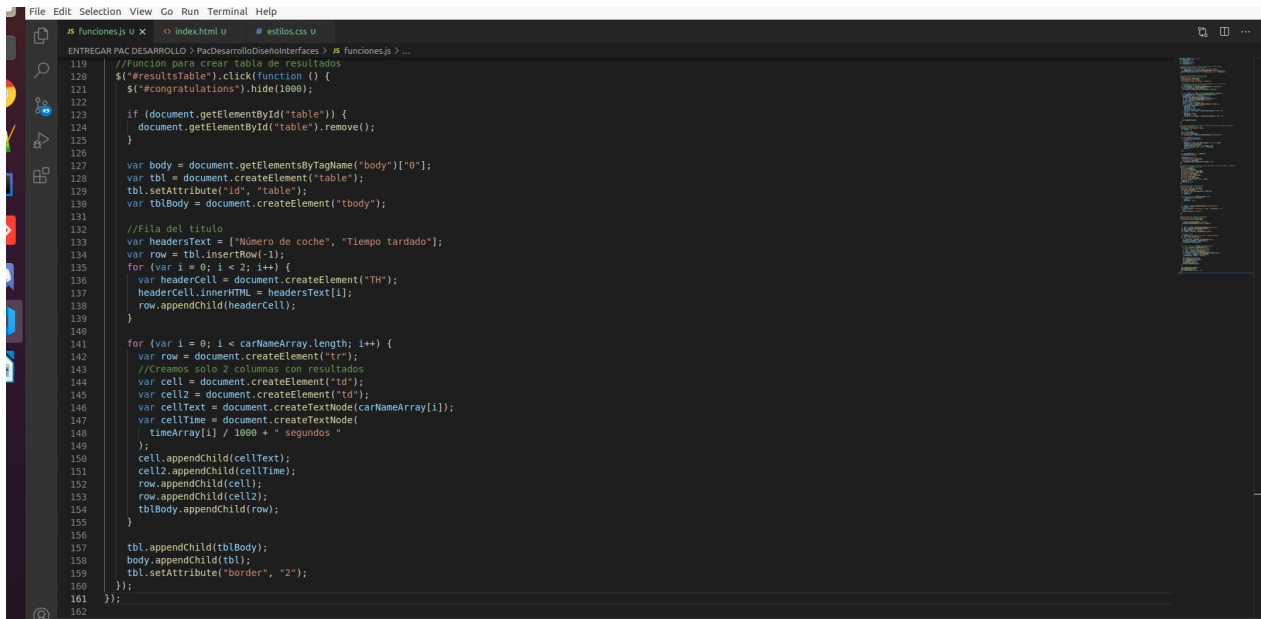
Figura 6: Código funciones.js IV

En la función para ver el coche ganador, usamos la función `min()` para sacar el valor mínimo del array en el que hemos almacenado los resultados.

Por último, para presentar la tabla hemos creado una función.

Esta función en primer lugar crea un elemento tabla, con el título de la misma y las celdas necesarias.

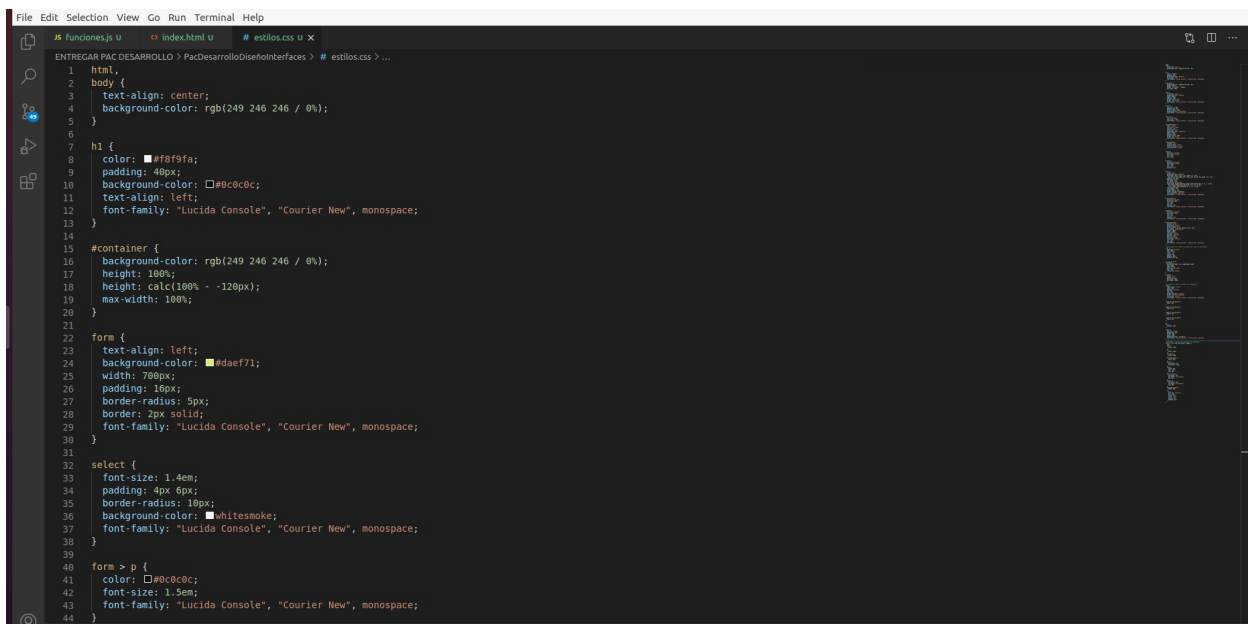
A continuación, se rellenan dichas celdas con los valores de los arrays que previamente hemos relleno.



```
119 //Función para crear tabla de resultados
120 $("resultsTable").click(function () {
121     $("#congratulations").hide(1000);
122
123     if (document.getElementById("table")) {
124         document.getElementById("table").remove();
125     }
126
127     var body = document.getElementsByTagName("body")[0];
128     var tbl = document.createElement("table");
129     tbl.setAttribute("id", "table");
130     var tblBody = document.createElement("tbody");
131
132     //Fila del título
133     var headersText = ["Número de coche", "Tiempo tardado"];
134     var row = tbl.insertRow(-1);
135     for (var i = 0; i < 2; i++) {
136         var headerCell = document.createElement("th");
137         headerCell.innerHTML = headersText[i];
138         row.appendChild(headerCell);
139     }
140
141     for (var i = 0; i < carNameArray.length; i++) {
142         var row = document.createElement("tr");
143         //Creamos solo 2 columnas con resultados
144         var cell = document.createElement("td");
145         var cell2 = document.createElement("td");
146         var cellText = document.createTextNode(carNameArray[i]);
147         var cellTime = document.createTextNode(
148             timeArray[i] / 1000 + " segundos "
149         );
150         cell.appendChild(cellText);
151         cell2.appendChild(cellTime);
152         row.appendChild(cell);
153         row.appendChild(cell2);
154         tblBody.appendChild(row);
155     }
156
157     tbl.appendChild(tblBody);
158     body.appendChild(tbl);
159     tbl.setAttribute("border", "2");
160 });
161
162
```

Figura 7: Código funciones.js V

Pasamos ahora a revisar el documento de estilos.css.



```
1  html,
2  body {
3      text-align: center;
4      background-color: rgb(249 246 246 / 0%);
5  }
6
7  h1 {
8      color: #f8f9fa;
9      padding: 40px;
10     background-color: #0c0c0c;
11     text-align: left;
12     font-family: "Lucida Console", "Courier New", monospace;
13 }
14
15 #container {
16     background-color: rgb(249 246 246 / 0%);
17     height: 100%;
18     height: calc(100% - 120px);
19     max-width: 100%;
20 }
21
22 form {
23     text-align: left;
24     background-color: #daeef7;
25     width: 700px;
26     padding: 10px;
27     border-radius: 5px;
28     border: 2px solid;
29     font-family: "Lucida Console", "Courier New", monospace;
30 }
31
32 select {
33     font-size: 1.4em;
34     padding: 4px 6px;
35     border-radius: 10px;
36     background-color: #whitesmoke;
37     font-family: "Lucida Console", "Courier New", monospace;
38 }
39
40 form > p {
41     color: #0c0c0c;
42     font-size: 1.5em;
43     font-family: "Lucida Console", "Courier New", monospace;
44 }
```

Figura 8: Código estilos.css I

En primer lugar se da estilo a los elementos básicos del archivo html (formulario, header, párrafo dentro del formulario para que tengan el mismo estilo...).

Seguidamente, pasamos a dar estilo a los elementos según el selector:

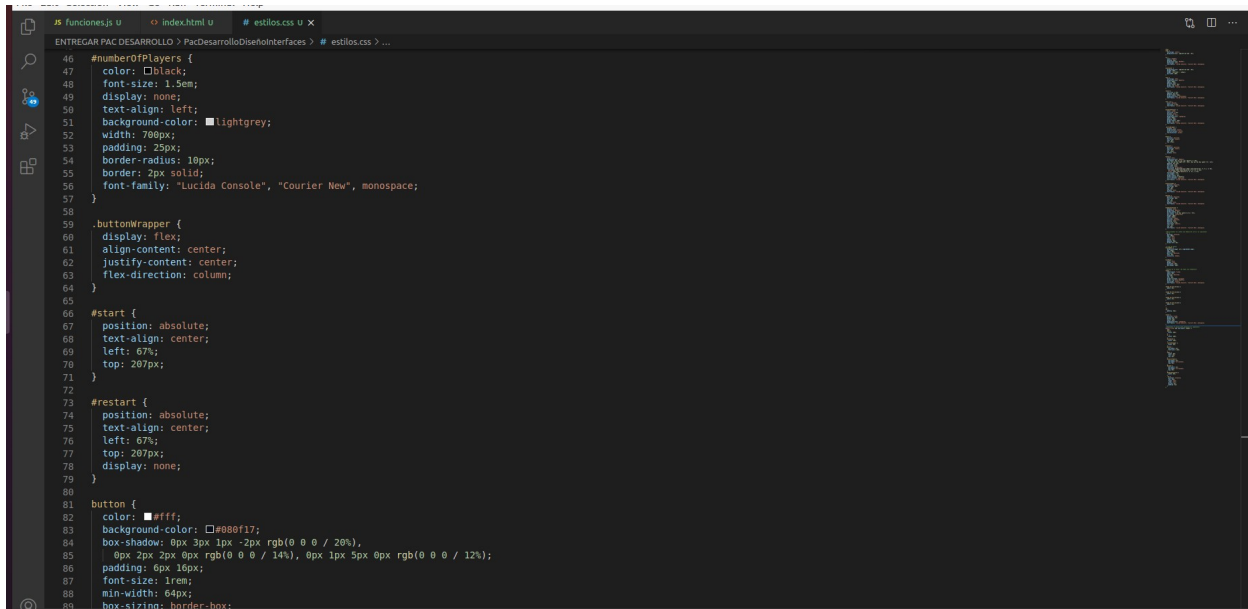


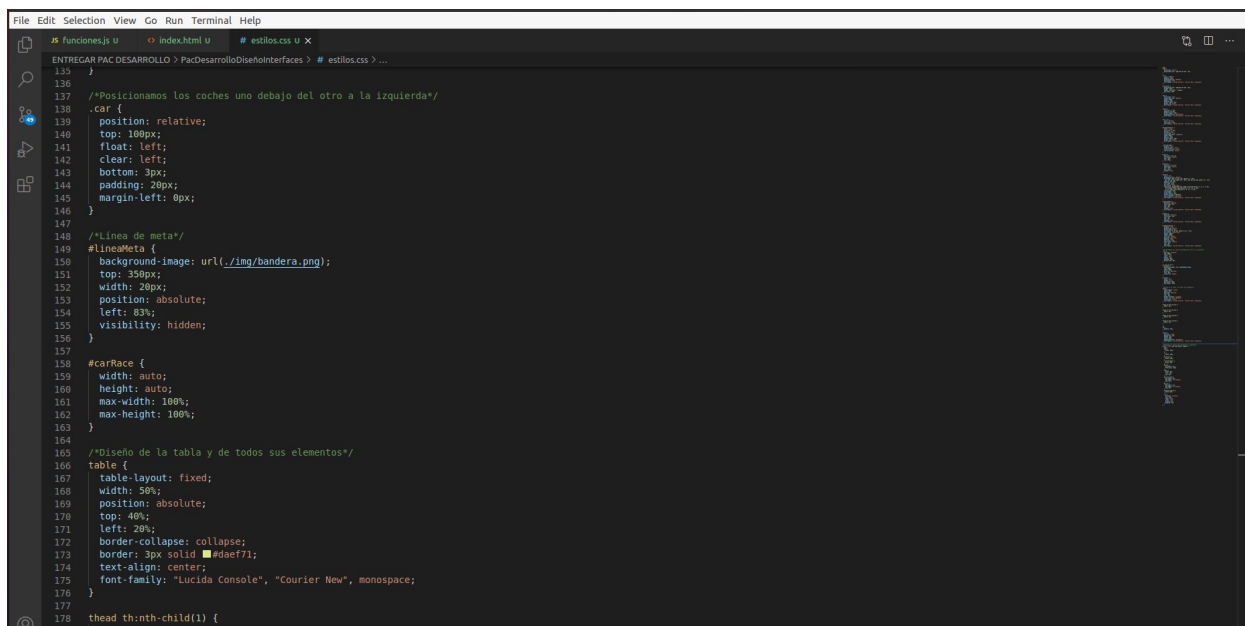
Figura 9: Código estilos.css II

Vemos que hay elementos a los que vamos a seleccionar en función de su clase (nomenclatura .class) , otros en función de su identificador (nomenclatura #identificador)...

De esta forma, todos los elementos dentro de la clase buttonWrapper por ejemplo, tendrán este mismo estilo, aunque luego, cada uno de ellos pueda tener otro estilo o modificación si se hace mención a su identificador.

En cuanto a las posiciones de los coches, queremos que salgan uno bajo el otro, por lo que indicamos una posición relativa a la izquierda con un margin-left de 0px.

Del mismo modo, la línea de meta (que a priori está oculta), se despliega una vez se selecciona la función que pide que se muestre. Hemos elegido la imagen de una línea de meta para este elemento.



```
File Edit Selection View Go Run Terminal Help
# funciones.js u index.html u # estilos.css u x
ENTREGAR PAC DESARROLLO > PacDesarrolloDiseñoInterfaces > # estilos.css > ...
135 }
136
137 /*Posicionamos los coches uno debajo del otro a la izquierda*/
138 .car {
139   position: relative;
140   top: 100px;
141   float: left;
142   clear: left;
143   bottom: 3px;
144   padding: 20px;
145   margin-left: 0px;
146 }
147
148 /*Linea de meta*/
149 #lineaMeta {
150   background-image: url(../img/bandera.png);
151   top: 350px;
152   width: 20px;
153   position: absolute;
154   left: 83%;
155   visibility: hidden;
156 }
157
158 #carRace {
159   width: auto;
160   height: auto;
161   max-width: 100%;
162   max-height: 100%;
163 }
164
165 /*Diseño de la tabla y de todos sus elementos*/
166 table {
167   table-layout: fixed;
168   width: 50%;
169   position: absolute;
170   top: 40%;
171   left: 20%;
172   border-collapse: collapse;
173   border: 3px solid #daef71;
174   text-align: center;
175   font-family: "Lucida Console", "Courier New", monospace;
176 }
177
178 thead th:nth-child(1) {
```

Figura 10: Código estilos.css III

En este documento también se pueden ver los detalles del diseño de la tabla y el footer (que hemos dispuesto como fijo con el atributo position:fixed).

Hemos decidido incluir un breve diseño responsive en el caso de que se visualice la aplicación en un monitor más pequeño.

BIBLIOGRAFÍA

- Material didáctico asignatura
- <https://www.mclibre.org>
- <https://developer.mozilla.org>
- <https://www.bufo.es/css>
- <https://francescricart.com>