Intro JS

INFORMACIÓN DE CONTACTO DEL INSTRUCTOR

Kevin Martínez - https://www.linkedin.com/in/kevinjmartinez/ - Discord Tag: kevinccbsg#2306

Ejercicios

Consideraciones generales

- **No se permite** el uso de librerías. Todo el código tiene que ser creado por el alumno.
- No es necesario crear un html para cada solución.

Ejercicio 1

Crea un archivo **ejercicio1.js** que tenga un objeto llamado usuario con los siguientes campos:

- Nombre (el tuyo o inventado)
- Apellidos (el tuyo o inventado)
- Una lista con los temas del bootcamp Node.js, Git y react con sus nombres y fechas de inicio de cada módulo. Fecha en formato “YYYY-MM-DD”
- Si estás en búsqueda activa con un valor de verdadero o false

En este archivo queremos mostrar por pantalla la fecha de inicio del módulo de react del objeto que hemos creado anteriormente.

Ejercicio 2 Arreglar bug

Nuestro cliente está intentando calcular el promedio de una lista de números pero nos dice que no funciona. No nos da el error, solo este código que es el que tiene en producción. Para este ejercicio tenemos que crear un archivo llamado **bug.js** con la solución.

```
const calcularPromedio = (numeros) => {
  let sumaTotal = 0;

  for (let i = 0; i <= numeros.length; i++) {
    sumaTotal += numeros[i];
  }

  const promedio = sumaTotal / numeros.length;
  return promedio;
};
const listaNumeros = [1, 2, 3, 4, 5];
const promedioNumeros = calcularPromedio(listaNumeros);
```

Ejercicio 3

En estos ejercicios no tienes acceso al enunciado. Debes deducir qué hacer observando los datos de entrada y el resultado:

3.1 Ejercicio

Crea una función para que con estos datos de entrada se produzca los siguientes resultados:

```
const input1 = [
  'Downloads',
  'Videos',
  'capture',
  'mp4',
]

// create your function here

yourFunction(input1); // 'Downloads/Videos/capture.mp4'

const input2 = [
  'CodinGame',
  'python',
  'py',
];

yourFunction(input2); // 'CodinGame/python.py'

const input3 = [
  'programming',
  'languages',
  'easy',
  'beginner',
  'useful',
  'pythonstuff',
  'py',
]

yourFunction(input3);
// 'programming/languages/easy/beginner/useful/pythonstuff.py'
```

3.2 Ejercicio

Crea una función para que con estos datos de entrada se produzca los siguientes resultados:

```
const input = 10;

// create your function here

yourFunction(input); // '1-0'

const secondInput = 1;

yourFunction(input); // '1'

const thirdInput = 11234;

yourFunction(input); // '1-1-2-3-4'
```

3.3 Ejercicio

Crea una función para que con estos datos de entrada se produzca los siguientes resultados:

```
const input1 = 'string';

// create your function here

yourFunction(input); // '6 gnirts'

const input2 = 'variable';

yourFunction(input); // '8 elbairav'

const input3 = 'pointer';

yourFunction(input); // '7 retniop'
```

Lo importante en estos ejercicios es ver el patrón con cada ejemplo. En ningún caso es necesario usar ningún tipo de condicional.

Ejercicio 4 Transformaciones con map y filter

Nuestro cliente tiene un array de datos y nos ha pedido que saquemos la siguiente información. El listado de los desarrolladores que tengan como habilidad “JavaScript” y el listado de los proyectos en el que sus desarrolladores trabajan.

Estos son los datos:

```
const datos = [
  {
    id: 1,
    nombre: 'Juan',
    habilidades: ['JavaScript', 'HTML', 'CSS'],
    proyectos: [
      { id: 1, nombre: 'Proyecto 1' },
      { id: 2, nombre: 'Proyecto 2' }
    ]
  },
  {
    id: 2,
    nombre: 'María',
    habilidades: ['Python', 'SQL', 'Django'],
    proyectos: [
      { id: 3, nombre: 'Proyecto 3' },
      { id: 4, nombre: 'Proyecto 4' }
    ]
  },
  {
    id: 3,
    nombre: 'Pedro',
    habilidades: ['Java', 'Spring', 'Hibernate'],
    proyectos: [
      { id: 5, nombre: 'Proyecto 5' },
      { id: 6, nombre: 'Proyecto 6' }
    ]
  }
];
```

Tenemos que hacer las operaciones necesarias para obtener estos 2 listados:

```
/* desarrolladoresJavascript */
[
  {
    "id": 1,
    "nombre": "Juan",
    "habilidades": ["JavaScript", "HTML", "CSS"],
    "proyectos": [
        { "id": 1, "nombre": "Proyecto 1"},
        { "id": 2, "nombre": "Proyecto 2" }
    ]
  }
]
/* nombresProyectos */
['Proyecto 1', 'Proyecto 2', 'Proyecto 3', 'Proyecto 4', 'Proyecto 5',
'Proyecto 6']
```

Hay que crear un archivo **transform.js** con la solución. Este archivo tiene que tener 2 funciones que nos retornen los valores correctos. NO USAR FOR NI WHILE. Se trata de un ejercicio para practicar el uso de map y filter.

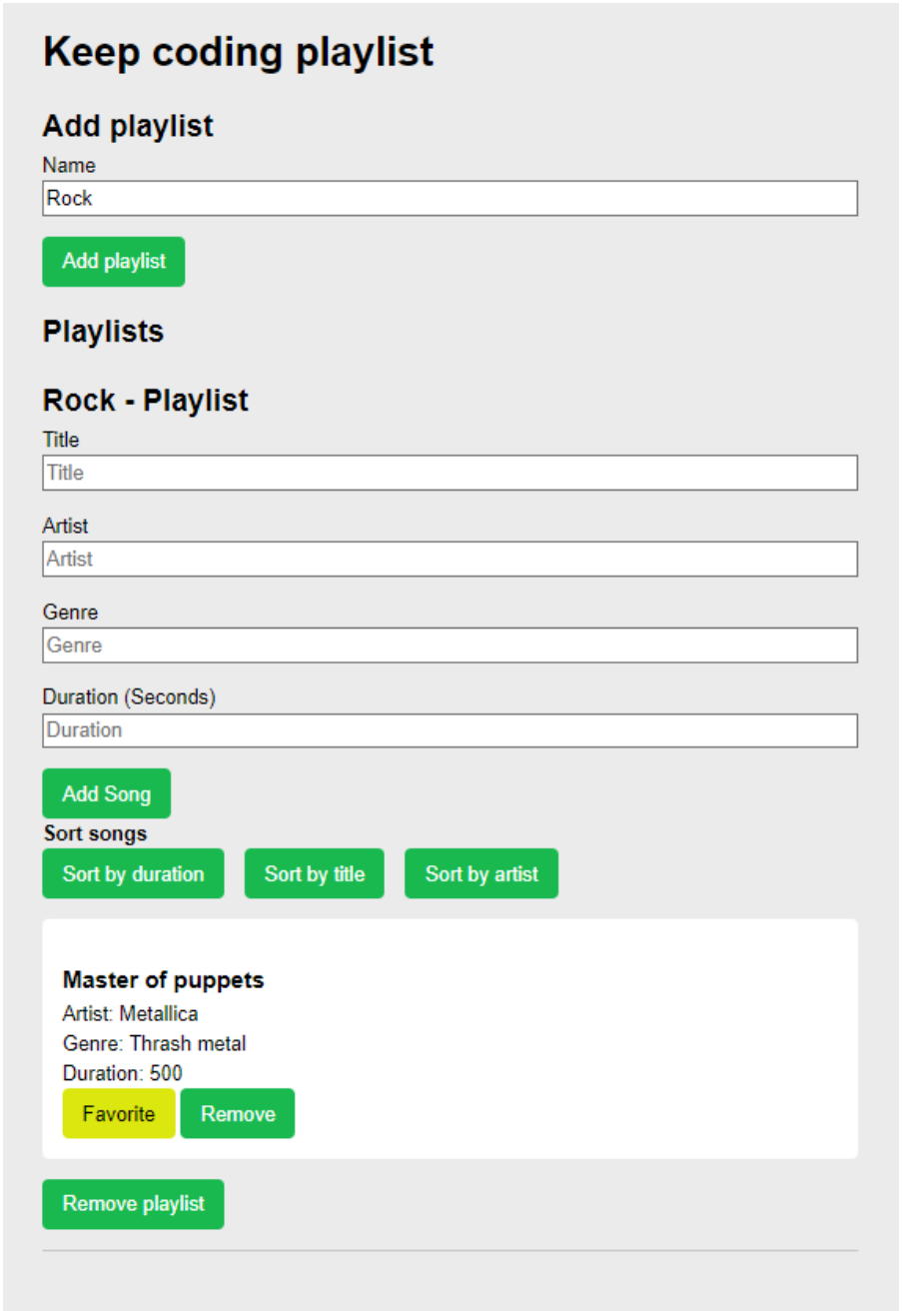
Ejercicio 5 Arreglar bug de asincronía

Tenemos otro error que nuestro cliente nos pide arreglar. El cliente está pidiendo un usuario y nos dice que está usando el id correcto el 1. Pero que siempre le da undefined. Nos ha pasado el código que tenemos que revisar y arreglar. Para este problema crear un archivo llamado **bugAsync.js** con la solución.

```
// Este programa simula una llamada asincrónica para obtener un usuario

function obtenerUsuario(id) {
  let usuario;

  setTimeout(() => {
    if (id === 1) {
      usuario = { id: 1, nombre: 'John Doe' };
    }
  }, 2000);

  return usuario;
}

const usuario = obtenerUsuario(1);
console.log(usuario);
```

Ejercicio 6: Keepcoding playlist

Tenemos un cliente que tiene una página donde se pueden crear listados de canciones. Es la siguiente:



El cliente ha creado la maquetación y la lógica de los formularios e interacciones con botones de para ordenar, marcar como favorito y borrar. Sin embargo, necesita nuestra ayuda para manejar el estado o lógica de esta web.

Para ello nos pide que creemos un estado en una función (usando cierres) que tenemos que exportar y se va a encargar de gestionar las funcionalidades de la app.

Nuestro cliente nos dio el archivo que tenemos que modificar para hacer funcionar su web. **Únicamente es necesario modificar este archivo**. Esté se encuentra en la carpeta **playlist/js/playlist.js**.

Cada playlist tiene un nombre y un listado de canciones. Las canciones tienen título, nombre del artista, género musical, duración en segundos y si es tú favorita.

En un comentario estaría la estructura de datos

```
/**
 * @typedef {Object} Song
 * @property {string} title - The title of the song.
 * @property {string} artist - The artist of the song.
 * @property {string} genre - The genre of the song.
 * @property {number} duration - The duration of the song in seconds.
 * @property {boolean} favorite - Whether the song is marked as a
favorite.
 */
// Example: { title: 'Song Title', artist: 'Song Artist', genre: 'Song
Genre', duration: 180, favorite: false }


/**
 * @typedef {Object} Playlist
 * @property {string} name - The name of the playlist.
 * @property {Song[]} songs - The list of songs in the playlist.
 */
// Example: { name: 'Playlist Name', songs: [{ title: 'Song Title',
artist: 'Song Artist', genre: 'Song Genre', duration: 180, favorite:
false }] }
```

A continuación os dejaré algunos pasos para ir empezando este reto. La web la podemos abrir usando live-server en la ruta **http://localhost:5500/playlist/playlist.html**

1. Abre el terminal de tu navegador con la web y revisa el error que aparece.
2. Tenemos la función pero no está exportada. El cliente quiere que se exporte como módulo por defecto. Recordemos como hicimos con el cart de clase `export default cart;`
3. Revisar que el error no aparece.
4. Ya tenemos conectado nuestro código a la web. Ahora toca implementar los métodos.
5. Empezar con “createPlaylist”. Este método tiene que añadir al listado de playlist un objeto playlist que tiene un “name” y un listado de canciones. Como no sabemos las canciones irá vacío.
6. Luego podemos implementar el “getAllPlaylists” que solo tiene que devolver las playlist.

7. En este punto se puede probar crear una playlist en la web y ver que aparece.
8. Podemos continuar con “removePlaylist”
9. Para el “addSongToPlaylist” vamos a recibir 2 parámetros, el nombre de la playlist para saber donde guardar el segundo parámetro que es un objeto con una canción. **IMPORTANTE: tener en cuenta que ese objeto no tiene el campo favorite.** Por lo que es necesario tener en cuenta al guardarlo ponerlo como false.
10. Con eso ya deberían salir canciones nuevas en cada playlist
11. Continuar con “removeSongFromPlaylist”
12. Opcional: método “favoriteSong” el cual a partir del nombre de la playlist y la canción hay que cambiar el favorite de false a true.
13. Opcional: método “sortSongs” a partir del nombre de la playlist y un criterio que pueden ser alguno de estos 3 valores 'title' | 'artist' | 'duration'. Ordenar el listado de canciones de esa playlist.
    a. title: Ordenar alfabéticamente
    b. artist: Ordenar alfabéticamente
    c. duration: Ordenar de menor a mayor

Recomendación: Haz commit cada vez que hagas uno de estos puntos y por cada ejercicio.
Ejemplo:

git commit -m “feat: add createPlaylist logic”