Instrucciones Generales

- Cada ejercicio debe ir en un archivo js por separado.
- Se puede subir la prueba a un repositorio público de su preferencia, en un .zip o en alguna plataforma online de Javacript (https://playcode.io/javascript-compiler por ejemplo, donde se generan urls públicas).

Ejercicio 1: Agrupar Nombres por Inicial

Descripción: Escribe una función que tome un array de cadenas (nombres) y devuelva un objeto en el que las claves sean las iniciales de los nombres y los valores sean arrays que contengan los nombres que empiezan con esa letra.

Instrucciones:

- Todos los nombres en el array serán cadenas de texto no vacías.
- Ignora la capitalización de las letras al agrupar.

Ejemplos:

```
function(["Ana", "Alberto", "Beatriz", "Bruno", "Carlos", "Clara"]);
//{
//  A: ["Ana", "Alberto"],
//  B: ["Beatriz", "Bruno"],
//  C: ["Carlos", "Clara"]
//}

function(["Mario", "Miguel", "marta", "Monica"]);
//{
//  M: ["Mario", "Miguel", "marta", "Monica"]
//}
```

Criterios de evaluación:

- Uso correcto de objetos y arrays.
- Manejo adecuado de strings.
- Claridad y eficiencia del código.

Ejercicio 2: Filtrar Palabras Repetidas

Descripción: Escribe una función que tome una cadena de texto que contiene varias palabras separadas por espacios y devuelva una nueva cadena donde se eliminen las palabras repetidas, manteniendo solo la primera aparición de cada palabra.

Instrucciones:

- La función debe ser insensible a mayúsculas y minúsculas, es decir, `"Palabra"` y `"palabra"` se consideran la misma palabra.
- La cadena resultante debe conservar el orden original de las palabras.

Ejemplos:

```
function("Este es un ejemplo ejemplo de prueba prueba");
// "Este es un ejemplo de prueba"

function("Hola hola mundo mundo Mundo");
// "Hola mundo"
```

Criterios de evaluación:

- Uso adecuado de arrays y objetos o conjuntos.
- Correcto manejo de strings.
- Eficiencia del código y claridad en la lógica.

Ejercicio 3: Calcular la Diferencia entre Fechas

Descripción: Escribe una función que tome dos cadenas de texto que representen fechas en formato `"YYYY-MM-DD"` y devuelva la diferencia entre ellas en días.

Instrucciones:

- Las fechas siempre estarán en el formato `"YYYY-MM-DD"`.
- La función debe devolver un número positivo si la primera fecha es anterior a la segunda, y negativo si la primera fecha es posterior.

Ejemplos:

```
function("2023-08-01", "2023-08-15"); // 14
function("2024-01-01", "2023-12-25"); // -7
function("2022-05-01", "2022-05-01"); // 0
```

Criterios de evaluación:

- Uso correcto de objetos `Date` en JavaScript.
- Se debe usar JavaScript, no está permitido utilizar librerías externas.
- Cálculo preciso de la diferencia en días.
- Claridad y eficiencia del código.

Ejercicio 4: Manejo de Promesas y Asincronía

Descripción: Escribe una función que haga una petición al siguiente endpoint `https://catfact.ninja/breeds` utilizando `fetch`, una vez recibido los datos debes hacer un filtrado por `pattern` y devolver un array de objetos ordenados por `pattern`.

Instrucciones:

- Utiliza `fetch` para realizar la petición.
- La función debe devolver un array de objetos ordenados por `pattern`.
- Puedes utilizar varias funciones para estructurar correctamente el código.
- Solo se debe tener en cuenta la primera página del API para ordenar.
- No se puede utilizar el método sort de los arrays

Ejemplos:

```
[
  {
    "breed": "Arabian Mau",
    "country": "Arabian Peninsula",
    "origin": "Natural",
    "coat": "Short",
    "pattern": ""
  },
  {
    "breed": "Bambino",
    "country": "United States",
    "origin": "Crossbreed",
    "coat": "Hairless/Furry down",
    "pattern": ""
  },
  {
    "breed": "American Curl",
    "country": "United States",
    "origin": "Mutation",
    "coat": "Short/Long",
    "pattern": "All"
  },
  {
    "breed": "Aegean",
    "country": "Greece",
    "origin": "Natural/Standard",
    "coat": "Semi-long",
    "pattern": "All"
  }
]
```

Criterios de evaluación:

- Correcta implementación de promesas y manejo de asincronía en JavaScript.
- Uso adecuado de funciones para ordenar estructuras de datos.
- Claridad en la estructura del código y en el manejo de las promesas.

Bonus:

Realizar un test con `jest` y `enzyme` de algunos de los ejercicios mencionados, no es necesario hacerlo de todos, con hacerlo para uno es suficiente. Este deberá tener el 100% cobertura de los tests unitarios de dicha función.