


Filter - Ejercicio

Crea un programa que muestre los empleador por salarios según la condición. Debes usar

Filter

js

 Copiar código

```
const Empleados = [

  {
    nombre: 'Julian Amador',
    cargo: 'Administrador de empresas',
    salario: 320000,
  },

  {
    nombre: 'Pedro Arango',
    cargo: '',
    salario: 1500000,
  },

  {
    nombre: 'Maria Diaz',
    cargo: 'Gerente',
    salario: 5000000,
  },

  {
    nombre: 'Natalia Sepulveda',
    cargo: 'Secretaria',
    salario: 1800000,
  },

  {
    nombre: 'Antonio Rodriguez',
    cargo: 'Ingeniero apoyo',
    salario: 2800000,
  }

]

// Creamos la constante con el filtro

const calcular = Empleados.filter(
  value => value.salario <= 2500000
)

//Finalizamos llamando la función calcular:
console.log(calcular);
```

Promise

`setTimeout` es una función en JavaScript que permite ejecutar otra función después de un cierto período de tiempo. Esto es útil para tareas como animaciones o actualizaciones de la interfaz de usuario que queremos realizar en el futuro sin interrumpir el flujo normal del código.

js

 Copiar código

```
setTimeout(() => {
  console.log("Hola desarrollador");
}, 4000
);


setTimeout(() => {
  console.log("Como estas?");
}, 6000
);

setTimeout(() => {
  console.log("Comencemos!");
}, 7000
);
```

Promise

Es un objeto que maneja operaciones asíncronas y permite gestionar su finalización o falla de manera más clara. Ayuda a evitar el anidamiento excesivo de callbacks y facilita el manejo de resultados asincrónicos.

js

 Copiar código

```
// Tenemos el siguiente arreglo
const Peliculas = [

  {
    nombre: 'Duro de matar',
    año: 1992,
  },

  {
    nombre: 'Ninja',
    año: 2009,
  },

  {
    nombre: 'La isla',
    año: 2013,
  },

  {
    nombre: 'La venganza',
    año: 2021,
  },

];

// Creamos la función
function getPeliculas(){
  return new Promise((resolve, reject) =>{ //Usando la promesa, 'reject' se usa en caso de que la
    if(Peliculas == 0){ // Validamos si mi arreglo no esta vacío
      reject(new Error ("No hay datos de las Peliculas"))
    }
    setTimeout(() =>{ resolve(Peliculas);}, 6000)
```

```
});  
}  
  
getPelículas()  
.then ((Películas) => console.log(Películas))  
.catch ((error) => console.log(err.message));
```


NOTA

Cuando se genera la promesa si se cumple la condición de lanza el `.then`. De lo contrario se lanza `.catch`

Ejercicio 2 sobre promesas

Vamos a ver otro ejercicio

js

 Copiar código

```
const Datos = [  
  
  {  
    Productos: "Pan",  
    precio: 1800,  
  },  
  
  {  
    Productos: "Huevos",  
    precio: 16000,  
  },  
  
  {  
    Productos: "Leche",  
    precio: 6000,  
  },  
  
];  
  
function getDatos(){  
  return new Promise((resolve, reject) => {  
    setTimeout(() => {  
      resolve(Datos);  
    }, 2500); // Corrección: cerrar paréntesis y agregar punto y coma aquí  
  });  
}  
  
getDatos().then((Datos) => console.log(Datos));
```

Async/Await

Permite escribir código asíncrono de manera más clara y fácil de entender. Permite que las funciones asíncronas se vean y se comporten como funciones síncronas, lo que hace que el código sea más legible y mantenible. Con Async/Await, puedes escribir código asíncrono de forma secuencial, evitando el anidamiento excesivo de callbacks o el encadenamiento de promesas. Esto simplifica la lógica de programación y hace que sea más sencillo manejar errores.

js

 Copiar código

```
function resolveAfter25Seconds(){
  return new Promise ((resolve) => {
    setTimeout(() => {
      resolve('resolved');
    }, 2000); // Corrección: ajustar el tiempo a 25000 milisegundos (25 segundos)
  });
}

asyncCall();

async function asyncCall(){
  console.log('calling');
  const result = await resolveAfter25Seconds();
  console.log(result);
  // Expected output: "resolved"
}
```

Ejercicio 1

js

 Copiar código

```
const Datos = [
  {
    Productos: "Pan",
    precio: 1800,
  },
  {
    Productos: "Huevos",
    precio: 16000,
  },
  {
    Productos: "Leche",
    precio: 6000,
  },
];

function getDatos() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve(Datos);
    }, 2500);
  });
}

async function RecibirDatos() {
  const D = await getDatos();
  console.log(D);
}


RecibirDatos();
```

NOTA

Para mayor claridad consulta el siguiente enlace [Click](#).

Ejercicio 2

js

 Copiar código

```
const EquiposF = [
  {
    Nombre: 'Cali',
    Ciudad: 'Barranquilla',
    AñoFundado: '',
  },
  {
    Nombre: '',
    Ciudad: '',
    AñoFundado: '',
  },
  {
    Nombre: '',
    Ciudad: '',
    AñoFundado: '',
  },
];

function getEquipos() {
  return new Promise((resolve, reject) => {
    if (EquiposF.length === 0) {
      reject(new Error("No hay datos en el arreglo"));
    }
    setTimeout(() => {
      resolve(EquiposF);
    }, 2500);
  });
}

async function mostrarDatos() {
  try {
    const equipo = await getEquipos();
    console.log(equipo);
  } catch (error) {
    console.error(error.message);
  }
}

mostrarDatos();
```