RETROALIMENTACIÓN

Primero se separan los eventos futuros de los pasados con un filtro.

```
   var futuros = eventos.filter(x => x.fecha.getTime() >
fechaReferencia.getTime())
   var pasados = eventos.filter(x => x.fecha.getTime() <=
fechaReferencia.getTime())
```

Luego se usa la función sort para ordenar el arreglo de futuros.

```
   futuros = futuros.sort((a,b)=>{
        if (a.fecha.getTime() > b.fecha.getTime()){
            return 1;
        }

        if (a.fecha.getTime() < b.fecha.getTime()){
            return -1;
        }

        return 0;
    })
```

Y puedes hacer lo mismo para los pasados, pero intercambiando 1 con -1 ya que el orden es inverso.

```
   pasados = pasados.sort((a,b)=>{
        if (a.fecha.getTime() > b.fecha.getTime()){
            return -1;
        }

        if (a.fecha.getTime() < b.fecha.getTime()){
            return 1;
        }
```

```
        return 0;
    })
```

Finalmente retornas los dos arreglos

```
    return [futuros,pasados];
```

El código completo debe quedar como se muestra a continuación:

```
function organizarEventos(eventos, fechaReferencia){
    var futuros = eventos.filter(x => x.fecha.getTime() >
fechaReferencia.getTime())
    var pasados = eventos.filter(x => x.fecha.getTime() <=
fechaReferencia.getTime())

    futuros = futuros.sort((a,b)=>{
        if (a.fecha.getTime() > b.fecha.getTime()){
            return 1;
        }

        if (a.fecha.getTime() < b.fecha.getTime()){
            return -1;
        }

        return 0;
    })

    pasados = pasados.sort((a,b)=>{
        if (a.fecha.getTime() > b.fecha.getTime()){
            return -1;
        }

        if (a.fecha.getTime() < b.fecha.getTime()){
            return 1;
        }

        return 0;
    })
```

```
    return [futuros,pasados];
}

var resultado = organizarEventos(eventos, new Date(2018, 3, 21));
console.log(resultado[0]);
console.log(resultado[1]);
```