Para todos los ejercicios

- Solamente se puede utilizar map, find, filter y reduce.
- No se puede utilizar ningún método salvo los mencionados.
- No se puede utilizar ninguna estructura de control (if, for, while, etc).
- Solamente tienes que escribir una línea de código.
- Los datos de los arrays pueden variar.

EJERCICIO 1

Se pide:
Nombre de los pueblos iberos con una población mayor de veintiuno. Una línea de código.

CÓDIGO STARTER:

```
<html>
<head>
<title>retos JavaScript</title>
</head>
<body>
<h1>retos JavaScript</h1>
<script>
let iberospoblacion =
[{pueblo:"Marmolejo",poblacion:55},{pueblo:"Bujalance",poblacion:50},
{pueblo:"Arjona",poblacion:10},
{pueblo:"Arjonilla",poblacion:20},{pueblo:"Porcuna",poblacion:25}];
let pueblos =
["Bujalance","Marmolejo","Marmolejo","Alharilla","Obulco","Lopera",
"Arjona", "Arjonilla","Porcuna"];
let iberos = ["Bujalance", "Arjona", "Arjonilla","Porcuna"];
//Nombre de las pueblos iberos con un población mayor de veintiuno
</script>
</body>
</html>
```

EJERCICIO 2

Se pide que hagas un programa que muestre la media de TODAS las notas de los 3 alumnos que tenemos en nuestro JSON.

CÓDIGO STARTER:

```
<!DOCTYPE html>
<html>
<body>
<p>
   Se pide que hagas un programa que muestre la media de TODAS las
notas de MOMAE de los alumnos que tenemos en nuestro JSON.
Ten en cuenta que voy a cambiar los datos del array para comprobar que
funciona tu programa.

</p>
<script>
var valumnos = {
"alumnos": [
{"nombre": "Irenita","notas": {"MOMAE": 8,"REDES": 0.71,
"ED": 7.04,"SSOO": 3.37,"DATABASE": 0.22,"PROG": 1.09,"FOL": 6.17}
},
{"nombre": "Marlaskito","notas": {"MOMAE": 7,"REDES": 0.71,"ED": 7.04,
"SSOO": 3.37,"DATABASE": 0.22,"PROG": 1.09,"FOL": 6.17}
},
{"nombre": "Bolañitos","notas": {"MOMAE": 6,"REDES": 0.71,"ED": 7.04,
"SSOO": 3.37,"DATABASE": 0.22,"PROG": 1.09,"FOL": 6.17}}
]
};

// ESCRIBE TU CÓDIGO AQUÍ

</script>
</body>
</html>
```

Se valorará de tu programa:

- El programa realiza la funcionalidad requerida.
- Que no haya errores de consola.
- La eficiencia.
- La modularidad (alta cohesión y bajo acoplamiento).

- Que el código sea limpio, flexible, reutilizable y mantenible.
- Sigue los principios SOLID.
- Otros criterios.