

Tarea 014 - JavaScript – Variante sobre Operaciones sobre Inventario de Productos

Los resultados de todas las tareas incorporarán, además del código fuente, los comentarios precisos y necesarios para su fácil comprensión. No escatimes esfuerzos en comentar el código, es una buena práctica para aprender, además de ser muy útil para modificaciones o reutilizaciones futuras.

Sería muy buena práctica añadir el título y el enunciado del ejercicio (como comentarios) al principio del código fuente.

Añade documentos en formato Word con capturas de la salida por pantalla (al ejecutar la página) si consideras que queda más clara la resolución del ejercicio.

Deberás entregar dos archivos con el resultado:

- 1. **Un documento en Word sin comprimir:** que contenga las imágenes que demuestren el resultado.
- 2. **Los archivos de código fuente:** en un único archivo, ya sea de extensión: **html, js, o css**. Si precisas entregar varios archivos, comprímelos en un único **zip**. (No admito rar).

El nombre del archivo entregado comenzará por tu nombre seguido por TareaXXX. Ejemplo: **federicoTarea014.zip**

Enunciado:

Partiendo del enunciado de la Tarea013, vamos a realizar las mismas operaciones, pero con teniendo en cuenta lo siguiente:

- La entrega debe tener un documento en Word (archivo de .docx, no se admite pdf) donde se vea el resultado y el código js. Pero que se vea bien, separa cada uno de los 4 apartados con su código y con su imagen de salida.
- Los resultados de los 4 apartados son los siguientes:

| | | | |
|---|--------|---|---|
| <pre>▼ ['Laptop'] ⓘ 0: "Laptop" length: 1 ▶ [[Prototype]]: Array(0)</pre> | 12.600 | <pre>▼ (4) [{...}, {...}, {...}, {...}] ⓘ ▶ 0: {nombre: 'Laptop', precio: 1650, cantidad: 4} ▶ 1: {nombre: 'Teléfono', precio: 550, cantidad: 10} ▶ 2: {nombre: 'Teclado', precio: 33, cantidad: 20} ▶ 3: {nombre: 'Monitor', precio: 220, cantidad: 5} length: 4 ▶ [[Prototype]]: Array(0)</pre> | <pre>▼ (2) ['Laptop', 'Teléfono'] ⓘ 0: "Laptop" 1: "Teléfono" length: 2 ▶ [[Prototype]]: Array(0)</pre> |
|---|--------|---|---|

- Resolver cada apartado de 2 formas, la primera de forma extensa y la segunda concisa.

A continuación, se muestra como ejemplo la resolución del primer apartado de las dos formas (extensa y concisa):

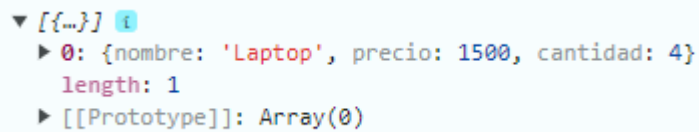
Forma extensa:

```
const inventario = [  
  { nombre: "Laptop", precio: 1500, cantidad: 4 },  
  { nombre: "Teléfono", precio: 500, cantidad: 10 },  
  { nombre: "Teclado", precio: 30, cantidad: 20 },  
  { nombre: "Monitor", precio: 200, cantidad: 5 }  
];  
  
// 1er paso: Función para saber si un producto vale más de 500 €  
function filtrarUnProductoCaro(producto){  
  if(producto.precio > 500){  
    return true;  
  } else{  
    return false  
  }  
}  
  
// 2º paso: función que obtiene los productos que cumplen la función anterior  
// Debes pasar la función filtrarUnProductoCaro como una referencia a filter,  
// sin paréntesis. Esto es porque filter se encargará de pasarle  
// cada elemento (producto) automáticamente a la función.  
function obtenerProductosCaros(arrayProductos){  
  const nuevoArray=arrayProductos.filter(filtrarUnProductoCaro);  
  return nuevoArray  
}
```

Si ahora se ejecutara la siguiente sentencia:

```
const nuevoArray=obtenerProductosCaros(inventario);
console.log(nuevoArray);
```

La salida por consola sería en este caso:



```
▼ [{...}] ⓘ
  ▶ 0: {nombre: 'Laptop', precio: 1500, cantidad: 4}
    length: 1
  ▶ [[Prototype]]: Array(0)
```

Donde se observa que es un array cuyos elementos (solo hay uno) son objetos del tipo producto. Sin embargo, el enunciado pide que se devuelva un array con los nombres de los productos únicamente.

Para ello vamos a utilizar **.map** que devuelve un array distinto con alguna transformación para cada elemento del array original. Antes definiremos la función que indique como queremos que sea la transformación de cada elemento del array original.

Añadimos al código anterior lo siguiente:

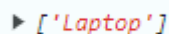
```
// 3er paso: función que transforma un producto en un nombre
function transformar(producto){
    return producto.nombre;
}

// 4º paso: función que transforma el array de elementos producto en array
// cuyos elementos son solo el nombre
function dejarSoloNombre(arrayOriginal){
    const nuevoArray = arrayOriginal.map(transformar);
    return nuevoArray;
}
```

Si ahora ejecutamos:

```
const nuevoArray1 = obtenerProductosCaros(inventario);
const nuevoArray2 = dejarSoloNombre (nuevoArray1);
console.log(nuevoArray2);
```

se obtiene la salida correcta:



```
▶ ['Laptop']
```

También hubiera funcionado igual si la ejecución se hiciera de golpe (sin utilizar nuevoArra1 y nuevArray2:

```
console.log(dejarSoloNombre(obtenerProductosCaros(inventario)));
```

Forma concisa:

A continuación se expone el ejemplo para el mismo apartado (el primero) pero utilizando funciones anónimas para **filter** y para **map**. Además, como son funciones simples, utilizaremos la notación arrow function

```
const inventario = [
  { nombre: "Laptop", precio: 1500, cantidad: 4 },
  { nombre: "Teléfono", precio: 500, cantidad: 10 },
  { nombre: "Teclado", precio: 30, cantidad: 20 },
  { nombre: "Monitor", precio: 200, cantidad: 5 }
];

// Todo de una vez
const obtenerProductosCarosConcisamente = arrayProductos => arrayProductos.
  filter(producto => producto.precio > 500).
  map(producto => producto.nombre)

console.log(obtenerProductosCarosConcisamente(inventario));
```

Con lo que obtendremos el mismo resultado que con la forma extensa.