Ejercicios JavaScript

Nacho Viano

Introducción a la Prog. con JavaScript – Ejercicios

1. **Ejercicio**
   Dadas dos variables numéricas A y B, que el usuario debe teclear, se pide realizar un algoritmo que intercambie los valores de ambas variables y muestre cuánto valen al final cada una de ellas (recuerda la asignación).

2. **Ejercicio**
   Algoritmo que lea dos números, calcule y escriba el valor de su suma, resta, producto y división.

3. **Ejercicio**
   a) Algoritmo que lea dos números y nos diga cuál de ellos es mayor o si son iguales (recuerda usar la estructura condicional if).
   b) Ahora con 3 números **diferentes**.

4. **Ejercicio**
   Diseñar un algoritmo que pida por teclado tres números; si el primero es negativo, debe imprimir el producto de los tres y si no lo es, imprimirá la suma.

5. **Ejercicio**
   Construir algoritmo tal que con un número entero como entrada, determine e imprima si es positivo, negativo o nulo.

6. **Ejercicio**
   Dado un número entero A, hacer un algoritmo que determine si es par, impar o nulo.
   Pista: para determinar el resto de una división, se usa la operación ***módulo %.***

7. **Ejercicio**
   Construir un algoritmo que dado el coste de un artículo vendido y la cantidad de dinero entregado, calcule e imprima el cambio que se debe entregar al cliente.

8. **Ejercicio**
   Dado el sueldo de un trabajador, diseña un algoritmo que aplique un incremento de sueldo del 15% si el sueldo es inferior a 1000€. Imprimir el nuevo sueldo.

9. Ejercicio

Construir un algoritmo que dado como datos 5 calificaciones de un alumno imprima el promedio y la palabra “aprobado” si el alumno tiene un promedio mayor o igual que 5, y la palabra “no aprobado” en caso contrario.

10. Ejercicio

Construir un algoritmo que dado la categoría y sueldo de un trabajador calcule el aumento de sueldo correspondiente teniendo en cuenta la siguiente tabla. Imprimir la categoría del trabajador y su nuevo sueldo.

CATEGORIA	INCREMENTO
1	15%
2	10%
3	6%
4	3%

11. Ejercicio

En una tienda efectúan un descuento a los clientes dependiendo de la cantidad de la compra. El descuento se basa en:

Si el monto es menor que 500€ -> No hay descuento
Si el monto está comprendido entre 500€ y 1.000€ **inclusive** –> 5% descuento
Si el monto está entre 1.000€ y 7.000€ **inclusive** -> 10% descuento
Si el monto está entre 7.000€ y 15.000€ **inclusive** -> 20% descuento
Más de 15.000€ -> 25% descuento

Imprimir (Escribir) el precio final.

12. Ejercicio

Construir un algoritmo que te permita calcular la temperatura teniendo en cuenta el número de sonidos emitidos por un grillo (en época de apareamiento) en un minuto, es una función que depende de la temperatura. Como resultado de esto, es posible determinar el nivel de temperatura haciendo uso de un grillo como termómetro.

La fórmula es:

T = **N / 4 + 40**, donde T es la temperatura en grados centígrados y N es el número de sonidos emitidos por minuto.

Como el aparato para medir los sonidos puede fallar, hay que tener en cuenta que si el número de sonidos es 0, es un error y debe de imprimir “error”.

13. Ejercicio

Construir un algoritmo tal que dado los datos de la base y la altura de un rectángulo calcule el perímetro y la superficie del mismo.

Superficie= base*altura
Perímetro = 2*(base + altura)

Comprobar los resultados con varios datos de entradas diferentes.

14. Ejercicio

Construir un algoritmo que resuelva el problema que tienen unos surtidores de gasolina, que registran lo que surten en galones, pero el precio de la gasolina se fija en litros. El algoritmo debe calcular e imprimir el precio que hay que cobrarle al cliente.

Precio gasolina = 1.333€/litro
1 galón = 3,78541 litros

15. Ejercicio

Modificar el algoritmo del ejercicio 6, de forma que, si se teclea un cero, se vuelva a pedir el número por teclado, así hasta que se teclee un número mayor que cero, recuerda la estructura **while**.

16. Ejercicio

Desarrollar un algoritmo que nos calcule el cuadrado de los 9 primeros números naturales.

17. Ejercicio

Se pide representar un algoritmo que nos calcule la suma de los N primeros números naturales. N se leerá por teclado.

18. Ejercicio

Se pide representar el algoritmo que nos calcule la suma de los N primeros números pares a partir de N. Es decir, si insertamos un 5, nos haga la suma de 6+8+10+12+14.

19. Ejercicio

Dada una secuencia de longitud indefinida de números leídos por teclado, que acabe con un –1, por ejemplo: 5,3,0,2,4,4,0,0,2,3,6,0,……,-1; Realizar el algoritmo que calcule la media aritmética. Suponemos que el usuario no insertará número negativos.

20. Ejercicio

Teniendo en cuenta que la clave es “eureka”, escribir un algoritmo que nos pida una clave. Solo tenemos 3 intentos para acertar, si fallamos los 3 intentos nos mostrara un mensaje indicándonos que hemos agotado esos 3 intentos. Si acertamos la clave, saldremos directamente del programa.

21. Ejercicio

Algoritmo que lea números enteros hasta teclear 0, y nos muestre el máximo, el mínimo y la media de todos ellos. Piensa como debemos inicializar las variables.

22. Ejercicio

Algoritmo que visualice la cuenta de los números que son múltiplos de 2 o de 3 que hay entre 1 y 100.

23. Ejercicio

Leer tres números que denoten una fecha (día, mes, año). Comprobar que es una fecha válida. Si no es válida escribir un mensaje de error y volver a pedir los números. Si es válida escribir la fecha cambiando el número del mes por su nombre. Ej. si se introduce 1 2 2006, se deberá imprimir “1 de febrero de 2006”. El año debe ser mayor que 0. (Recuerda la estructura **switch**).

24. Ejercicio

Calcular las calificaciones de un grupo de alumnos. La nota final de cada alumno se calcula según el siguiente criterio: la parte práctica vale el 10%; la parte de problemas vale el 50% y la parte teórica el 40%. El algoritmo leerá el nombre del alumno, las tres notas, escribirá el resultado y volverá a pedir los datos del siguiente alumno hasta que el nombre sea una cadena vacía. Las notas deben estar entre 0 y 10, si no lo están, no imprimirá las notas, mostrará un mensaje de error y volverá a pedir otro alumno.

25. Ejercicio

Algoritmo que lea un número entero (lado) y a partir de él cree un cuadrado de asteriscos con ese tamaño. Los asteriscos sólo se verán en el borde del cuadrado, no en el interior.
Ejemplo, para lado = 4 escribiría:

```
* * * *
*     *
*     *
* * * *
```

Recuerda la estructura repetitiva ***For.***

26. Ejercicio

Desarrollar un algoritmo que lea 10 números por teclado y calcule el cubo de cada uno de ellos. En cada lectura, tiene que indicar por pantalla el número que está pidiendo. Ejemplo de salida por pantalla:

*Introduce el número **1º**.*
➢ 8
El cubo de 8 es 512.
*Introduce el número **2º**.*

27. Ejercicio

Desarrollar un algoritmo que imprima la tabla de multiplicación del número N introducido por teclado. Para N = 13, el *output* sería:

13 X 1 = 13
13 X 2 = 26
…
13 X 10 = 130

28. Ejercicio

Desarrollar un *timer* o temporizador. La cantidad de segundos con la que se quiere hacer la cuenta atrás se introducirá por teclado. Cuando llegue al final, se imprimirá *"¡¡Ring!!"* y el programa acabará.

- Investigar cómo hacer los intervalos en JS

29. Ejercicio

Desarrollar una calculadora de factoriales para números introducidos por teclado.

*El factorial de un número **N** es la multiplicación de todos los números desde 1 hasta N. Es decir, para N = 5, el factorial de 5 sería: **5! = 5*4*3*2*1 = 120***

30. Ejercicio

Escribir un algoritmo que muestre por pantalla un triángulo como los siguientes hasta un número de filas introducido por teclado.

a) Para filas = 4

```
1
12
123
1234
```

b) Para filas = 4

```
1
22
333
4444
```

c) Para filas = 4

```
1
2 3
4 5 6
7 8 9 10
```

31. Ejercicio

Algoritmo que lea un número entero (altura) y a partir de él cree una escalera invertida de asteriscos con esa altura. Deberá quedar así, si ponemos una altura de 5.

```
*****
 ****
  ***
   **
    *
```

32. Ejercicio

El siguiente es el menú de un restaurante de bocadillos. Diseñar un algoritmo capaz de leer el número de unidades consumidas de cada alimento ordenado y calcular la cuenta total. Vamos a suponer que estos precios son fijos, es decir, que son constantes (recuerda que en PSeInt no se usa comas para separar la parte decimal de la parte entera).

PRODUCTO	PRECIO
Bocadillo de Jamón	1,5€
Refresco	1,05€
Cerveza	0,75€
Pan	2€

33. Ejercicio

Crear un array de tamaño 10 y que guardará números enteros introducidos por teclado. Tras introducirlos todos, imprimirá cada índice junto con el valor al que corresponda.

34. Ejercicio

a) Generar un número aleatorio (del 1 al 10) que el usuario debe adivinar.
b) Aumentar el límite superior a 100 y añadir una ayuda al usuario: escribir si el número es mayor o menor que la lectura.

35. Ejercicio

Crear un array de números donde le indicaremos el tamaño por teclado. Rellenará cada elemento con números aleatorios entre 0 y 9. Posteriormente, mostrará por pantalla el valor de cada posición junto con su índice y finalmente, la suma de todos los valores.

36. Ejercicio

Crear dos arrays de números enteros de longitud 10 rellenos con números aleatorios del 1 al 20. Imprimir índice y el resultado de la multiplicación de ambos elementos de los arrays del índice de cada iteración. *Cuidado con los elementos del array sin inicializar.*

37. Ejercicio

Una calculadora de la letra de un DNI, pediremos el DNI por teclado y nos devolverá el DNI completo con la letra. Para calcular la letra, cogeremos el resto del DNI entre 23, que será un número entre 0 y 22. Utilizar el resultado para buscar en un array de caracteres la posición que corresponda a la letra. Esta es la tabla de caracteres:

Posicion	Letra	11	B
0	T	12	N
1	R	13	J
2	W	14	Z
3	A	15	S
4	G	16	Q
5	M	17	V
6	Y	18	H
7	F	19	L
8	P	20	C
9	D	21	K
10	X	22	E

38. Ejercicio

Dado un array de números de 5 posiciones con los siguiente valores [1, 2, 3, 4, 5], guardar los valores de este array en otro array distinto pero con los valores invertidos, es decir, que el segundo array deberá tener los valores [5,4,3,2,1].

39. Ejercicio

Dado dos arrays del mismo tamaño, determinar, elemento a elemento, si ambos son iguales. Con que un elemento sea diferente, se considerarán los arrays como diferentes. Escribir al final del algoritmo el resultado.

40. Ejercicio (difícil)

Generar un array de 20 elementos con números aleatorios no repetidos entre sí.

41. Ejercicio

Dado un array de N números enteros que se generen aleatoriamente, hacer un algoritmo que:

a) Obtenga cuántos números son mayores que 0.
b) Calcule el promedio de los números positivos.
c) Obtenga el promedio de todos los números.

43. Ejercicio

Partir del ejercicio 28 pero esta vez realizar un reloj digital completo que nunca pare. Tendrá la estructura **horas:minutos:segundos**. Ejemplo de salida: 23:15:39

***Nota**: deberás utilizar "Esperar" y "Limpiar pantalla".*

44. Ejercicio

Partir del ejercicio 2 y añadir la siguiente funcionalidad:
El usuario tendrá un menú numérico en pantalla para poder elegir entre las operaciones a realizar. Si da una opción incorrecta (no existe), el programa avisará al usuario y volverá a mostrar el menú. Hará esto hasta que el usuario elija la opción de salir del programa. Ejemplo de menú impreso por pantalla:
"Seleccione el número de una de las siguientes opciones:
1: Sumar
2: Restar
3: Multiplicar
4: Dividir
5: Salir del programa."

45. Ejercicio

Teniendo un vector con los números naturales que queramos, meter en otro de la misma longitud, aquellos que sean pares y mayores que 25.

Después, mostrar el vector de origen completo y el de destino solo los números introducidos.

46. Ejercicio

Dados A, B y C que representan a números enteros diferentes construir un algoritmo para escribir estos números de forma descendente.

47. Ejercicio

Dados un array de 5 elementos con números aleatorios del 1 al 100. Imprimir el estado inicial del array, ordenarlo de forma ascendente y volver a imprimir el nuevo estado.

48. Ejercicio

Almacenar una lista de nombres en un array y luego ordenarlos alfabéticamente. Para la entrada de datos se utiliza una estructura Mientras, sin saber a priori la cantidad de datos que se ingresarán.

Pista: Los datos alfanuméricos (strings) también se pueden comparar con los operadores < y >.

49. Ejercicio

Se tienen los costes de producción de tres departamentos (dulces, bebidas y conservas) correspondientes a los 12 meses del año anterior. Construir algoritmo que proporcione:

a) ¿En qué mes se registró el mayor coste de producción de dulces?
b) Promedio anual de los costes de producción de bebidas
c) ¿En qué mes se registró el mayor coste de producción en bebidas, y en qué mes el menor coste?
d) ¿Cuál fue el que tuvo menor coste de producción en diciembre?

50. Ejercicio

Hacer un algoritmo que cuente las veces que aparece una determinada letra en una frase que introduciremos por teclado.

51. Ejercicio

Comprobar si un número N positivo es primo. Se dice que un número entero positivo N es un número primo si los únicos enteros positivos que lo dividen son exactamente 1 y N (él mismo).

52. Ejercicio

Rellenar un array con 10 números aleatorios entre 1 y 15. Posteriormente, buscar un número introducido por teclado y nos debe decir si está incluido en el array y el índice de su primera coincidencia.

53. Ejercicio

Usar una función para calcular el promedio recibiendo un array y su longitud. En el algoritmo principal, leer la cantidad de datos que introducirá el usuario y posteriormente los propios datos. Escribir el resultado de su promedio.

54. Ejercicio

Diseñar un algoritmo que lea el número N e imprima y cuente todos los números primos de 2 hasta N.

55. Ejercicio

Generar una matriz de 4 filas y 5 columnas con números aleatorios entre 1 y 100. Imprimirla en forma de matriz o tabla, con sus filas y columnas.

56. Ejercicio

Leer y guardar en una matriz las notas de los alumnos de un colegio en función del número de cursos (filas) y del número de alumnos por curso (columnas). El máximo de alumnos será 5 para cada uno de tres cursos.

57. Ejercicio

Dada un matriz cuadrada A (array de 2 dimensiones con el mismo número de columnas que de filas, por ejemplo: 3) construir un algoritmo que permita determinar si dicha matriz es simétrica. Se considera que una matriz es simétrica si A[i,j] = A[j,i] para todos los elementos i,j de la matriz.

58. Ejercicio

Crear un array de 3 páginas, 4 filas y 5 columnas donde el primer elemento valga 1, el segundo 2, el tercero 3 y así sucesivamente, e imprimirla.

59. Ejercicio

Se dispone de un array de 5 páginas, 4 filas y 10 columnas, que se refieren al centro, al curso y al número de alumnos de un colegio respectivamente. Imprimir la nota media por curso y la nota media máxima y su centro de pertenencia.

Ejercicio 60 - Bonus

Desarrollar una calculadora de potencias y factoriales usando funciones recursivas.

Ejercicio 0 – Sintaxis Básica

Los apartados se pueden resolver de la manera que nos resulte más sencilla siempre que sigamos las posibles restricciones que se definan. Deben realizarse en un mismo archivo javascript separándolos con comentarios de forma clara. Deberemos tener cuidado con redifinir variables / constantes con el mismo nombre. Cuando no sepamos la respuesta a un apartado por nosotros mismos, se puede (y debe) investigar en la chuleta y en internet. Es obligatorio comprobar por consola el correcto estado de las variables y el correcto funcionamiento de las funciones de cada apartado.

1. Define e inicializa un array con 5 elementos *string* **en una sola línea.**

2. Define un array inicialmente **vacío**. Añade tres elementos de tipo ***number*** posteriormente. Elimina por completo el primero y añade dos nuevos números al inicio. En cada paso, imprime la longitud y el array entero por consola utilizando un *string template* del tipo*:* `Longitud: ${}`.

3. Escribe una función nombrada que devuelva ***true*** si el argumento dado es de tipo *boolean* y *false* en caso contrario.

4. Escribe una función que devuelva la longitud de un *string* recibido por argumento.

Ejercicio 0 – Sintaxis Básica

5. Crea una función de flecha que reciba una cantidad de minutos y lo devuelva convertido en segundos.

6. Crea una función que reciba un número y devuelva el siguiente número par. (Si él es par, lo devolverá directamente).

7. Crea una función que reciba una edad y devuelva la cantidad de días que esa persona ha vivido. Puedes obviar los años bisiestos.

8. Crea una función que reciba un array y devuelva su último elemento. Prueba tu función con varios arrays de diferentes longitudes.

9. Un granjero necesita contar la cantidad de patas de los animales en sus diferentes granjas muy frecuentemente, pero tiene tres especies: pollos (2 patas), vacas (4 patas) y cerdos (4 patas). Tu función recibirá la cantidad de animales en ese orden y devolverá la cantidad de patas totales. Ejemplo:

```
countLegs(5, 2, 8); // output: 50
```

Ejercicio 0 – Sintaxis Básica

10. Crea una función que determine si dos datos recibidos por argumentos son del mismo tipo.

11. Crea una función que reciba un *string* con una frase y devuelva un array donde cada elemento será una palabra de la frase original. *Investigar método existente de los strings para este fin.*

12. Inicializa dos objetos, *address1* y *address2* con las propiedades: **provincia, ciudad, municipio, código postal, calle, número, planta,** y **número de puerta.**

13. Los dominios en la web, se componen del nombre de dominio (releevant) y de un TLD (top-level domain) como, por ejemplo .com / .es / .org, etc. Crea una función que se llame **parseDomain()** que reciba por argumento un *string* y devuelva un objeto con dos propiedades: domain y tld. Ejemplo:

```
parseDomain("releevant.com");
/* output:
{
    domain: “releevant",
    tld: "com"
}
*/
```

Ejercicio 0 – Sintaxis Básica

14. Nos han prohibido el uso del operador de igualdad estricta (===), pero queremos poder seguir utilizando dicha esa funcionalidad. Crea una función que devuelva *true* si dos números tienen el mismo valor y el mismo tipo de dato. Debemos usar el operador lógico "&&". Prueba tu función con dos estos inputs:

```
strictEquality("5", 5); // false
strictEquality(5, 5); // true
```

15. Crea una función que reciba dos strings y devuelva *true* si tienen la misma longitud y *false* en caso contrario.

16. Crea una función que reciba un *string* y determine si está vacío **sin utilizar la propiedad *length.***

17. Imprimir **elemento a elemento** el array del apartado 1 de cuatro formas diferentes:
    - while
    - for
    - for of
    - forEach.

Ejercicio 0 – Sintaxis Básica

18. Crea una función que reciba un *string* y un número N y devuelva el *string* original repetido N veces.

```
repeatString("No haré memes sobre el profesor. ", 2);
// output: ”No haré memes sobre el profesor.  No haré memes sobre el profesor. ”
```

19. Crea una función que recibe un objeto con dos campos, votos positivos y votos negativos y que devuelva la cuenta final.

```
getVoteCount({upVotes: 35, downVotes: 15}); // 20
```

20. Crea una función que recibe un array de tipos de datos mezclados y que devuelva otro array con el tipo de cada uno de los elementos.

```
getTypes(["I'm learning JS in a Bootcamp 🚀", 7.5, {}, 0, undefined, [], "releevant"]);
// output ["string", "number", "object", "number", "undefined", "object", "string"];
```

Ejercicio 0 – Sintaxis Básica

21. Función que dado un array de números con formato *string* devuelva un array con los números ya *parseados*.

```
getParsedNumbers(["1.5", "10", "0"]); // output: [1.5, 10, 0];
```

22. Crea una función de flecha que devuelva “Positivo” si el número que recibe por argumento es mayor o igual que cero y “Negativo” en caso contrario. **Usa el operador ternario.**

23. Crea una función que dado un array cualquiera y un índice, borre el elemento guardado en ese índice.

24. Usando la función del apartado anterior, crea otra función que dado un array de números y un número a filtrar, devuelva un array borrando todos las apariciones de dicho número.

```
filterNumber([1, 5, 6, 7, 5], 5); // output: [1, 6, 7]
```

25. Crea dos funciones que recibirán un objeto, la primera devolverá un array con los nombres de todas sus propiedades. La segunda devolverá un array con los **valores** de dichas propiedades.
*Investigar los métodos **keys** y **values** del prototipo de Object.*

Ejercicio 0 – Sintaxis Básica

26. Crea una función que invierta un *string*.

```
stringReverse(".nóicamargorp ed sedrat sal ne llub der led érasuba oN");
// output: No abusaré del red bull en las tardes de programación.
```

27. Crea una función que compare *strings* sin tener en cuenta las mayúsculas / minúsculas.

```
compareStrings("Darth CODER", "darth coder"); // output: true
```

28. Crea una función que convierta en mayúscula sólo la primera letra de cada palabra de un string dado. El apartado 11 será de ayuda. *Investigar cómo unir un array de strings en un único string.*

```
capitalize("comprobaré los errores de la consola antes de pedir ayuda.");
// output: "Comprobaré Los Errores De La Consola Antes De Pedir Ayuda."
```

29. Crea una función en **una única línea** que reciba un valor lógico y devuelva el opuesto.

Ejercicio 1 - DOM

1. Partimos de un HTML con un div vacío. Con JS, añadir dos elementos p con un texto dentro.

2. Al pulsar un botón, cambiar el color del fondo del cuerpo de HTML.

3. Partimos de un HTML con una lista de 3 URLs (texto) de imágenes y un element img. Al hacer click en cada URL, cambiará la imagen a la que contenga dicha URL.

4. Añadir un input de tipo texto y un botón. Al pulsar el botón debe escribirse el texto del input en un párrafo.
   4.2 Añadir un nuevo input pero esta vez cambiará el texto con cada pulsación de tecla del usuario.

5. Similar al anterior, pero será para un textarea y validará si lo introducido es mayor de 15 caracteres.

6. Añadir un input de tipo texto con leyenda: “Escribir un número par”. Añadir un botón. Al pulsar el botón nos validará si el número es par o no. En caso negativo, cambiar los bordes del input a rojo.
   Para revertir el estado de una propiedad, podemos utilizar el valor “revert” o dejarla vacío.

7. Partiendo de una lista **ul**, crear 10 **li** con un texto indicando el número del elemento (“Elemento X”) usando un bucle for.

Ejercicio 1 (cont.)

8. Crear un enlace y un botón. Si pulso el enlace se me abre la URL en la misma página. Si pulso primero el botón y luego el enlace, se me abre en una nueva pestaña.

9. Añadir un párrafo y un **select** con 5 opciones de colores: negro, blanco, rojo, amarillo, verde y azul. Al seleccionar un color del select, cambiar el color del párrafo.

10. Incluir un botón que al pulsarlo genere un número aleatorio y mantenga en una lista actualiza el número de elementos que se han generado, los que son pares y los que son impares.

11. Construir una lista que tenga números. Añadir un input donde poder añadir números y un botón. Al pulsar el botón, si el número ya existe en la lista, mostrar un mensaje de error, si no existe, lo añadirá al principio.

12. Crearemos una clase .btn en CSS que le de ciertos estilos a un botón. Al hacer click en el botón haremos “toggle” o alternaremos esa clase, es decir, si está presente la quitaremos y si no está, la añadiremos.

Ejercicio 1 (cont.)

El código siguiente, añade un *eventListener* a cada botón para que cuando se haga click en cada uno de ellos, le cambie el backgroundColor.

Refactorizar el código para hacerlo con un único forEach.

Nota:
1. Para transformar un HTMLCollection a un array, podemos hacer: ***Array.from(HTMLCollection)****;*

2. Para acceder al elemento que "disparó" el evento, podemos usar ***evento.target****.*

```
const buttons = document.getElementsByClassName('btn');

buttons[0].addEventListener('click', () => {
    buttons[0].style.backgroundColor = "red";
});

buttons[1].addEventListener('click', () => {
    buttons[1].style.backgroundColor = "red";
});

buttons[2].addEventListener('click', () => {
    buttons[2].style.backgroundColor = "red";
});
```

Ejercicio 2

Vamos a crear una lista que muestre toda la información de un usuario. Habrá un selector para poder elegir los diferentes usuarios que tenemos guardados en un array. Al seleccionar uno diferente actualizará la lista para mostrar la información del usuario que esté seleccionado.

Los usuarios serán objetos y contendrán las siguientes propiedades y métodos:

- Nombre, Primer apellido, Segundo apellido, Email, Edad, Ciudad, Número de artículos en carrito de compra, Función que incremente en 1 el número de artículos y Función que vacíe el carrito.

Añadir dos botones fuera de la lista, uno que incremente de uno en uno los artículos del carrito y el otro que los vacíe. La lista será generada con un bucle *for...in* sobre las propiedades del objeto para obtener el nombre y los valores (pares de key y values).

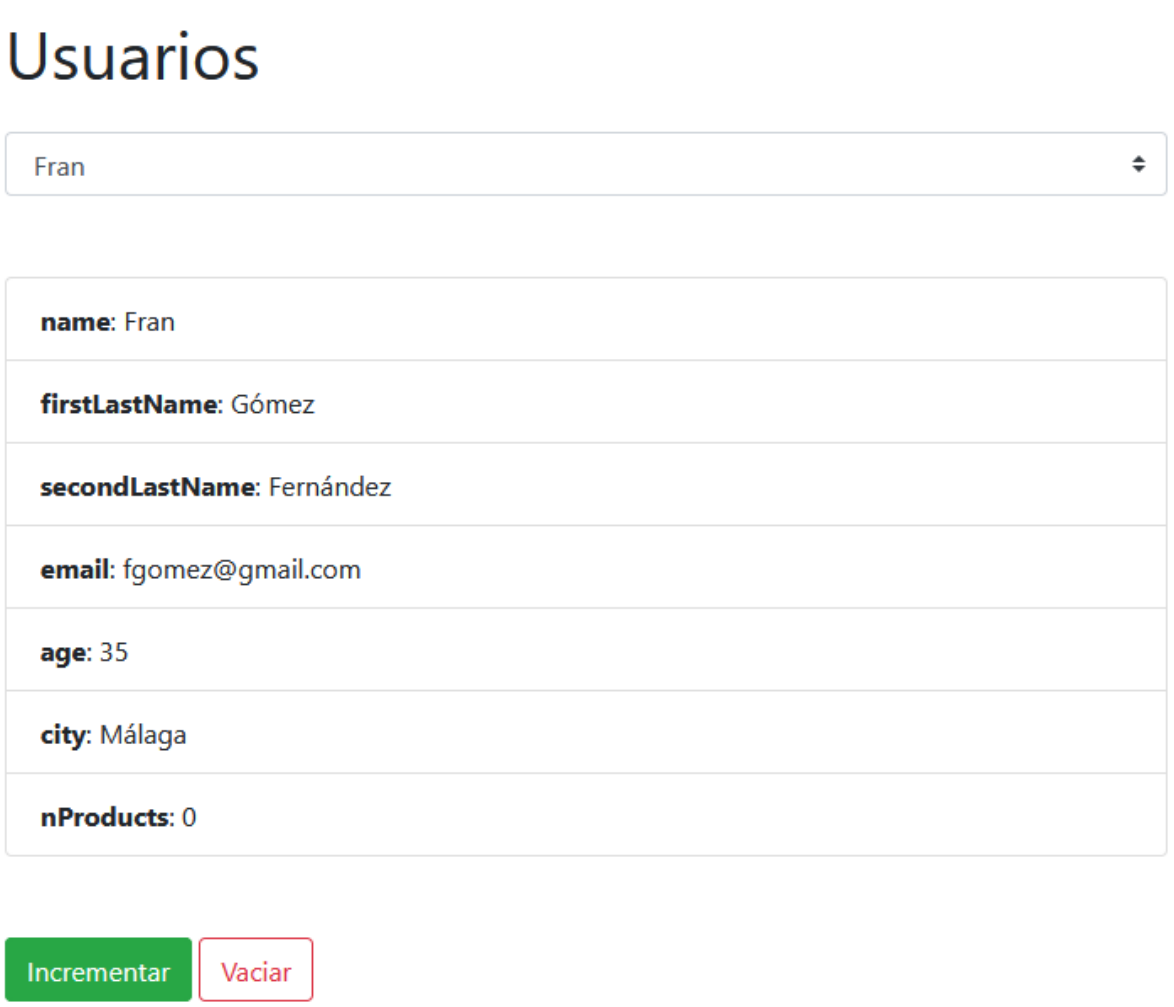
Ejercicio 2 (cont.)

Posible hoja de ruta:

1. Crear el constructor del usuario con sus propiedades y métodos.
2. Rellenar el array con varios usuarios.
3. Creamos el HTML y CSS necesarios y accedemos desde JS a los elementos necesarios del DOM y guardamos las referencias.
4. Rellenamos el selector con el nombre de cada usuario.
5. Rellenamos la lista con todas las propiedades de un usuario cualquiera.
6. Añadir *listeners* necesarios.
7. Si el selector cambia, actualizamos la lista con toda su información.
8. Si pulsamos los botones, tendrá que actualizarse el número de productos del usuario seleccionado.

Ejercicio 2 (cont.)

Aspecto final aproximado usando Bootstrap 4:



Ejercicio 3

Arrays y objetos

1. Generar un array de 100 objetos que tengan las siguientes propiedades:
   - **Cargo:** construido con un string plantilla (`${}`) cuyos valores sean “Empleado 1”, “Empleado 2”, etc.
   - **Rendimiento:** un número aleatorio con 2 decimales entre 0 y 1.
   - **Salario:** un número aleatorio entre 1250 y 4000.

2. Ordenar el array por rendimiento e imprimirlo. Usar una función anónima como *callback*.

3. Ordenar el array por salario e imprimirlo. Usar una función de flecha.

4. Ordenar el array por el número de empleado de forma decreciente. El número de empleado sólo estará **dentro** del string **Cargo**. Usar una función nombrada como *callback*.

5. Usando filter: imprimir el cargo y salario de los que cobren más de 2500€.

6. Usando forEach: subir el sueldo un 25% a los que cobren menos de 1500€.

Ejercicio 3 (Cont.)

Arrays y objetos

7. Usando map: Imprimir un array de objetos que tengan sólo la propiedad salario.

8. Usando reduce: Obtener el coste total de todos los sueldos para la empresa teniendo en cuenta que a la empresa le cuesta tener un empleado su sueldo más un 15% por impuestos.

9. Usar el método o métodos (reduce / map / filter / sort) que determinemos oportuno e imprimir en cada apartado:
   - Despedir a los que tengan un rendimiento menor a 0.3.
   - Calcular el sueldo medio de la empresa.
   - Subir el sueldo de los que tengan un rendimiento superior a 0.7.

Ejercicio 4

Crear una tabla que muestre un array de libros guardado en memoria. Requisitos:

1. Usar estilos de Bootstrap 4. Se puede utilizar el CSS directamente desde el CDN.

2. Crear un constructor de objeto Book con los siguientes atributos: **ID, título, autor, ventas** y **precio.**

3. Rellenar un array con 10 libros.

4. Para cada elemento del array, tendremos crear una nueva fila en la tabla con las columnas para todas las propiedades del objeto además de un botón para poder eliminar el libro.
   Se aconseja la creación de una función para actualizar la tabla (o el tbody) desde cero.

5. Si el botón de eliminar es pulsado, borrará el libro del array y actualizará la tabla.

6. Añadir un pequeño formulario después de la tabla para poder anexar un nuevo libro a nuestro array. Tras hacer *submit*, además de añadirlo, deberá actualizar la tabla y vaciar los *inputs*.

Ejercicio 4 (cont.)

Aspecto final aproximado usando clases de Bootstrap 4:

Books

#	Título	Author	Sales	Price	Remove
1	The Selfish Gene	Richard Dawkins	740120	12	Eliminar
2	The God Delusion	Richard Dawkins	610120	15	Eliminar
3	La nueva mente del emperador	Roger Penrose	120000	17	Eliminar
4	Sapiens: A Brief History of Humankind	Yuval Noah Harari	910120	16	Eliminar
5	The Selfish Gene	Richard Dawkins	740120	12	Eliminar

Add a new Book

Title

Enter title

Author

Enter author

Ventas

Enter sales

Precio

Enter price

Save

Ejercicio 5

DOM

1. Realizar una tabla filtrable. Tendremos un input de búsqueda y una tabla de libros rellena por JavaScript. Cada vez que cambie el input, se actualizará la tabla para que aparezcan sólo los libros cuyos títulos contengan lo que estamos introduciendo en el input. Partiremos del ejercicio 4 en lo que nos haga falta.

2. Añadiremos un botón para ordenar la tabla por el precio de forma creciente / decreciente. (Efecto toggle).

3. Añadiremos una última fila en tfoot, separada del resto, que nos sume los precios de los libros que están presentes en la tabla en ese momento.

Ejercicio 6

Fetch API + DOM – Paginación en el lado del cliente

1. Haremos una petición GET a esta API y guardaremos todos los Posts. Haremos una paginación en nuestro array, de forma que se muestren sólo 20 artículos en el DOM con un h1 para *title* y un p para el *body*.

   Al mostrarse sólo 20 artículos, tendremos que “crear” tantas páginas como sea necesario para poder mostrar el número total de *posts* en diferentes páginas.

   Podremos ir navegando por las páginas con un par de botones de anterior / siguiente que actualicen los *posts* y se mostrarán de 20 en 20 hasta que no queden más. Tendremos que tener cuidado con no pasarnos por delante ni por detrás del array.

Ejercicio 6

Fetch API + DOM – Paginación en el lado del servidor

2. Hacer un login sencillo que haga una petición POST a https://reqres.in/api/login con las credenciales (investigar qué objeto espera recibir). Si tenemos éxito, mostraremos una **lista** con todos los usuarios de este otro punto de la API: https://reqres.in/api/users. Si no tiene éxito, mostrará una alerta avisando con el error.

   Como tiene múltiples páginas, tendremos que ir solicitando todas las páginas de una en una hasta que no haya más páginas. El parámetro para obtener una página específica es **?page=2**. Éste se pondrá al final de la URL. Se recomienda hacer las peticiones en un bucle e ir concatenando los usuarios ya sea a un array o directamente al DOM. Considerar hacerlo con async / await. Más información de la API en https://reqres.in

Ejercicios JavaScript

Nacho Viano