EXAMEN 3ª EVALUACIÓN – DAW Dual

*Nombre:*_____ ***Fecha: 07/04/24***

Debes entregar los 2 **archivos** en una carpeta con tu nombre : **apellido_nombre**. Los nombres de los ficheros deben ser:

- **apellido_nombre_ejercicio1.html**
- **apellido_nombre_ejercicio2.html**

Se penalizará:

- Que el código no esté correctamente formateado y comentado.
- Que los ficheros no estén nombrados adecuadamente.
- *Que se utilice código Javascript dentro de los elementos HTML. Todo el código debe estar entre las etiquetas <script> o en un fichero independiente.*
- *Dejar un ejercicio en blanco.*

Recursos

Cómo seleccionar otros nodos según algunas de las relaciones de parentesco establecidas alrededor de tal elemento.

- **parentNode :** Por medio de **parentNode** podemos seleccionar el elemento padre de otro elemento.
- **firstChild:** Con **firstChild** lo que seleccionamos es el primer hijo de un elemento. Por desgracia, hay discrepancias entre los diversos navegadores sobre qué debe considerarse o no hijo de un nodo, por lo que esta propiedad en ocasiones complica demasiado un script.
- **lastChild:** La propiedad **lastChild** funciona exactamente como **firstChild**, pero se refiere el último de los hijos de un elemento. Se aplican, por tanto, las mismas indicaciones anteriores.
- **nextSibling:** Gracias a nextSibling, lo que podemos seleccionar es el siguiente hermano de un elemento. Se aplican las mismas limitaciones que para las dos propiedades anteriores.
- **previousSibling : previousSibling** funciona igual que nextSibling, pero selecciona el hermano anterior de un elemento.

Creación de elementos

- **appendChild:** Por medio de appendChild podemos incluir en un nodo un nuevo hijo, de esta manera:

  elemento_padre.appendChild(nuevo_nodo);

  El nuevo nodo se incluye inmediatamente después de los hijos ya existentes —si hay alguno— y el nodo padre cuenta con una nueva rama.
- **insertBefore:** Nos permite elegir un nodo del documento e incluir otro antes que él.

  elemento_padre.insertBefore(nuevo_nodo,nodo_de_referencia);
- **insertAfter:** No hay un metodo que inserte un nodo detrás de otro

- **replaceChild:** Para reemplazar un nodo por otro contamos con replaceChild, cuya sintaxis es: elemento_padre.replaceChild(nuevo_nodo,nodo_a_reemplazar);
- **removeChild:** Dado que podemos incluir nuevos hijos en un nodo, tiene sentido que podamos eliminarlos. Para ello existe el método removeChild. elemento_padre.removeChild(nodo_a_eliminar);
- **cloneNode:** Por último, podemos crear un clon de un nodo por medio de cloneNode: elemento_a_clonar.cloneNode(booleano);
  El booleano que se pasa como parámetro define si se quiere clonar el elemento —con el valor false —, o bien si se quiere clonar con su contenido —con el valor true—, es decir, el elemento y todos sus descendientes.

Método find:

```
const array1 = [5, 12, 8, 130, 44];
const found = array1.find((element) => element > 10);
console.log(found);
```

Array.prototype.map():

```
const numbers = [1, 2, 3, 4];
const filteredNumbers = numbers.map((num, index) => {
  if (index < 3) {
    return num;
  }
});
```

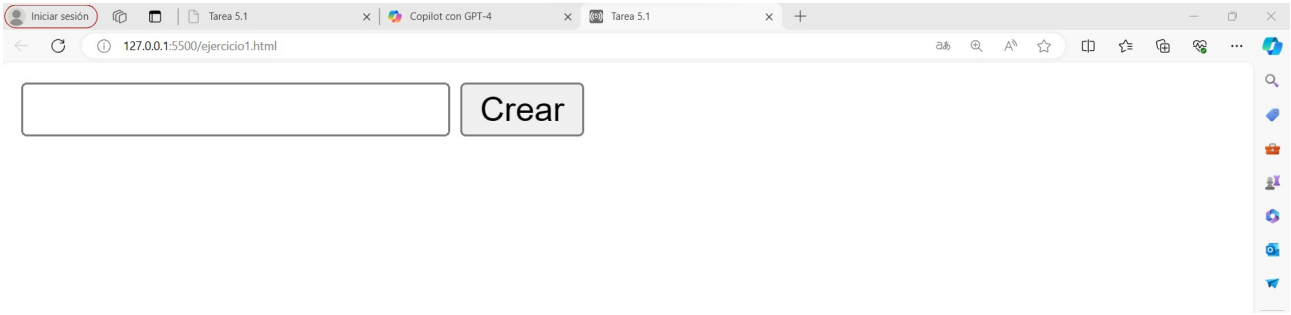
Modificar estilos

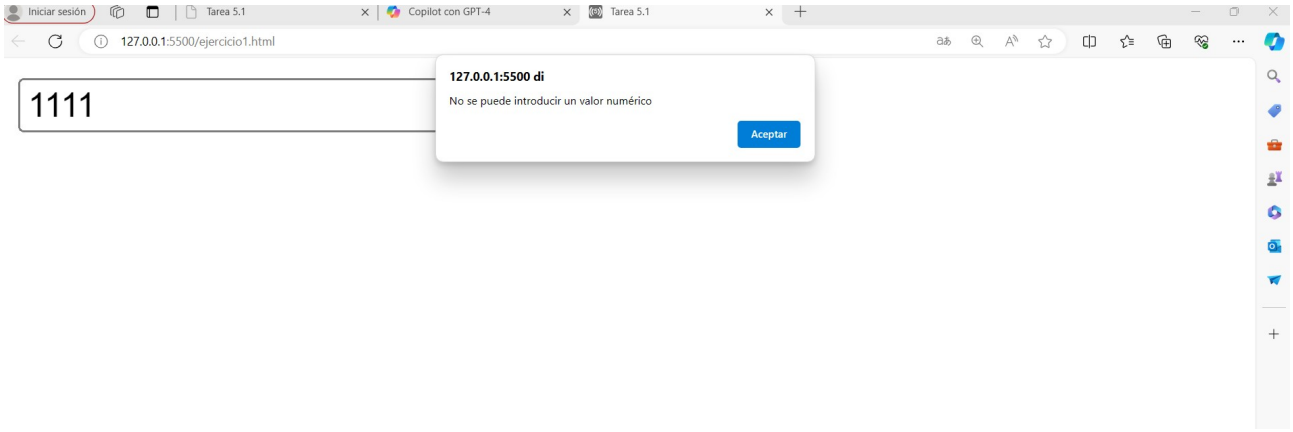
```
miElemento.style.backgroundColor
miElemento.style.fontWeight
miElemento.style.textDecoration=”line-through”
miElemento.style.color
```

1. **Ejercicio 1: Crea un programa JavaScript que se ejecute en el navegador que muestre el el siguiente aspecto y funcionalidad (4 puntos)**

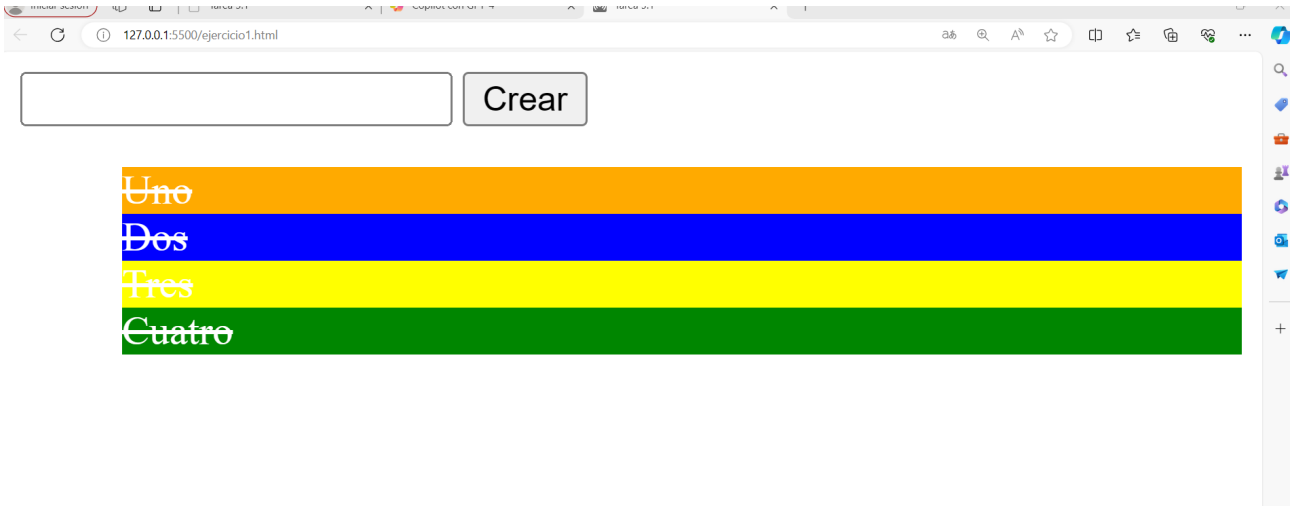
   Funcionamiento:

   - **Apartado 1:** El usuario introduce un valor en el cuadro de texto. Al pulsar el botón crear comprueba si es numérico. En caso de que no lo sea crea una nueva viñeta con el contenido. Si el valor es numérico muestra un alert. En cualquiera de los dos casos se debe limpiar el cuadro de texto y asignar el foco.





   - **Apartado 2:** Al pulsar el botón derecho sobre cada elemento de la lista modifica el aspecto. El color de fondo se debe generar a través de una función de forma aleatoria.

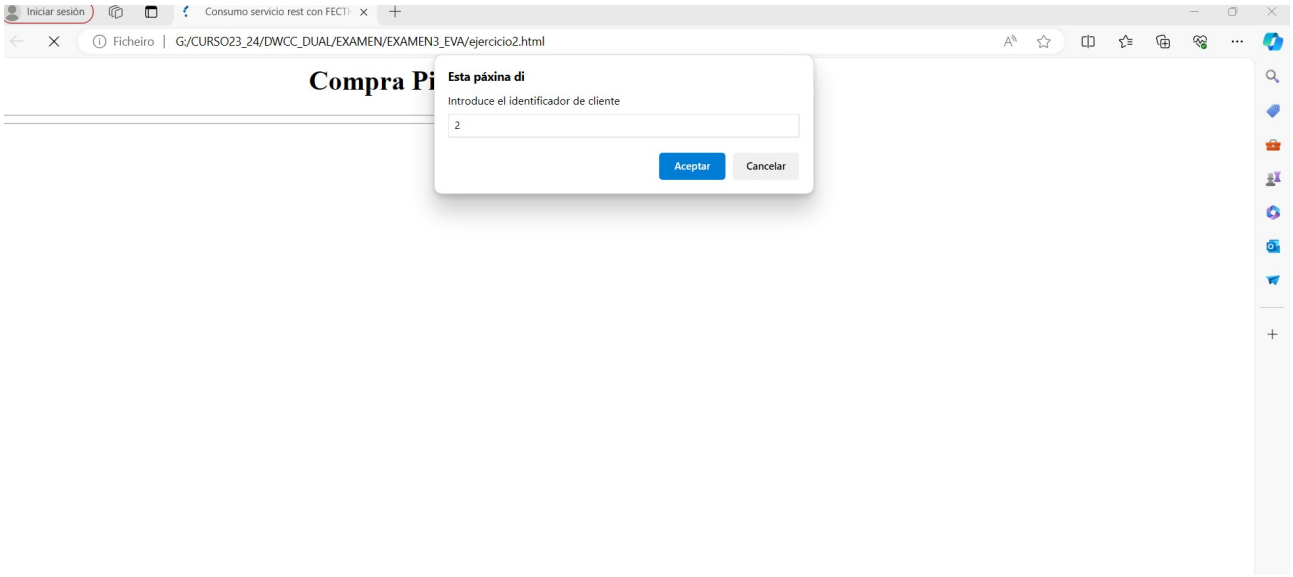


- **Apartado 3:** Al hacer clic sobre cada elemento de la lista, lo elimina
- **Apartado 4:** Funcionalidad con delegación de eventos, funciones, sintaxis flecha, declaración de variables

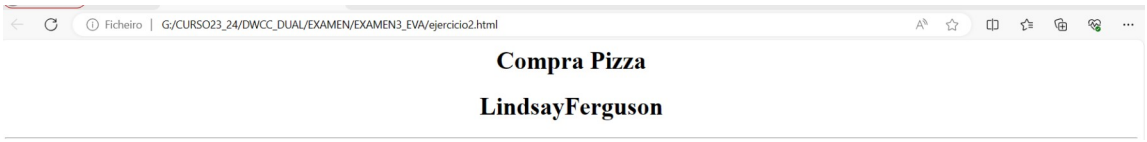
- **Apartado 1: (1 punto)**
- **Apartado 2: (1 punto)**
- **Apartado 3: (1 punto)**
- **Apartado 4: (1 punto)**

2. **Ejercicio 2: Crea un programa JavaScript que se ejecute en el navegador que muestre el el siguiente aspecto y funcionalidad (6 puntos)**

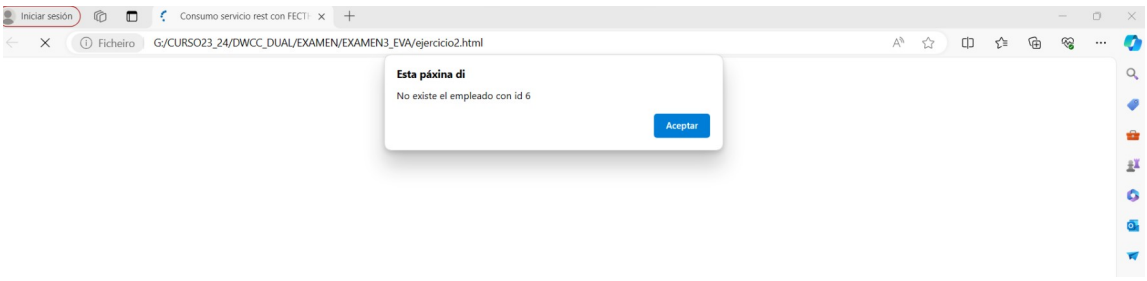
   - **Funcionamiento:**
     - **Apartado 1:** El usuario introduce un identificador de cliente con prompt.



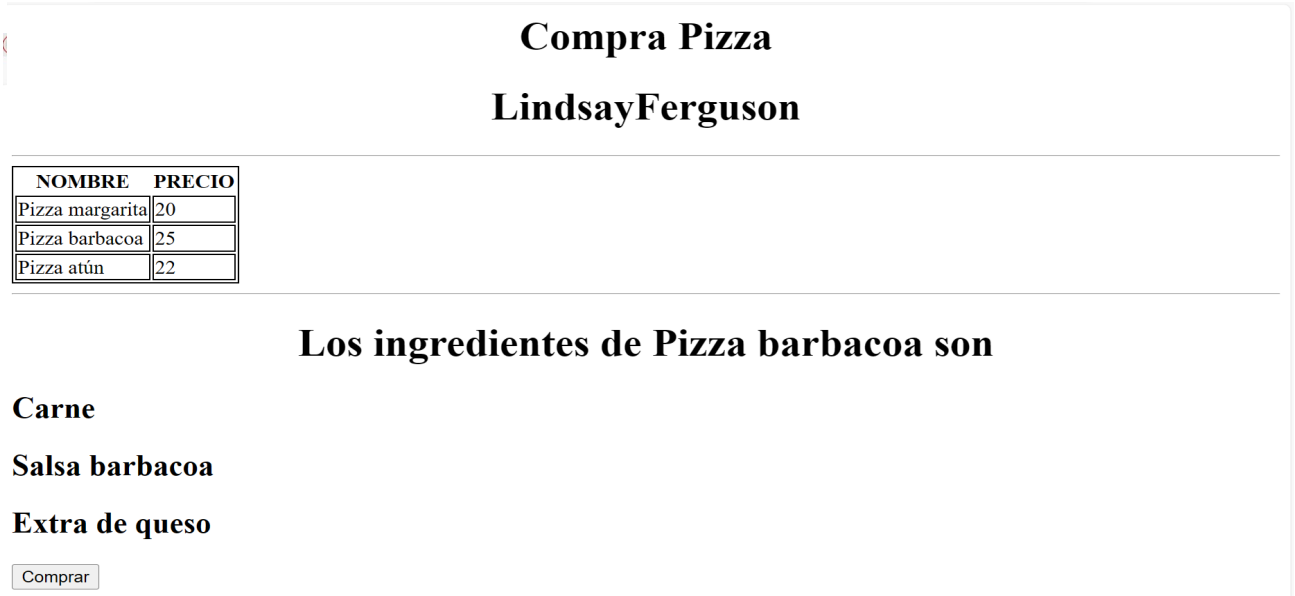
Recorre el array de clientes. Si lo encuentra muestra el nombre creando un nodo de tipo H1, con su nombre. Debes crear una función para mostrar los datos del cliente.



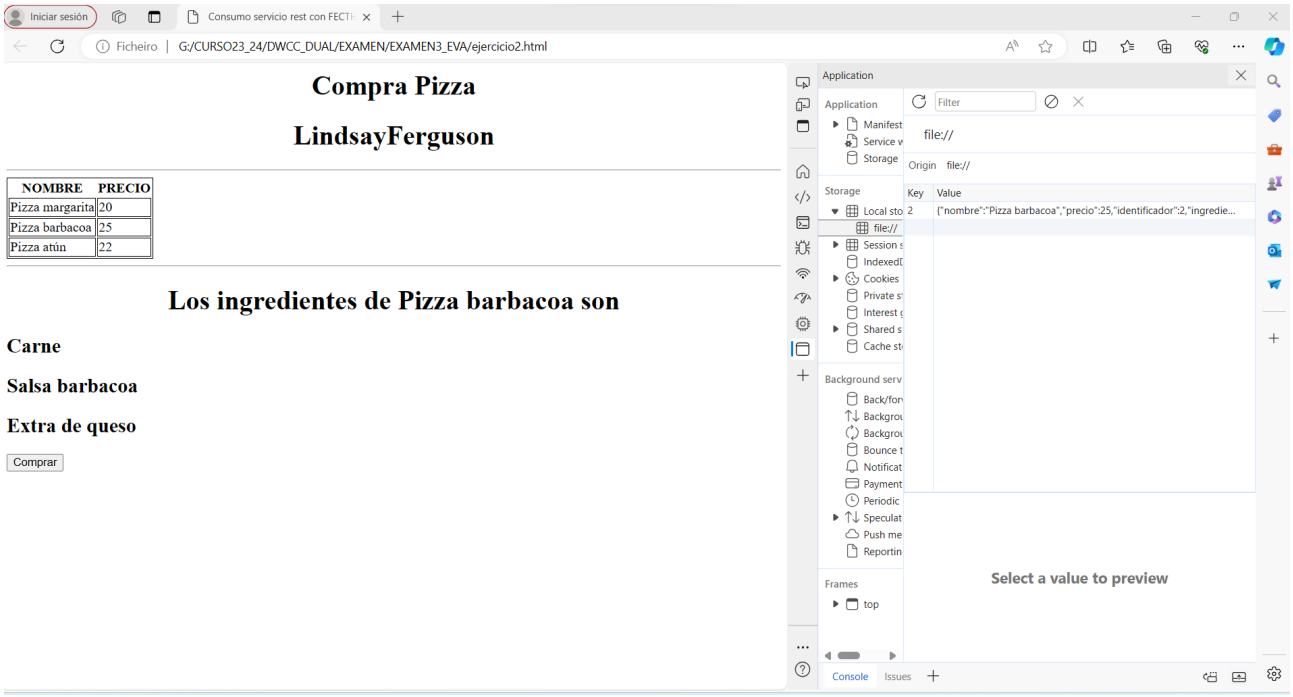
     - En caso de que no lo encuentre muestra un **alert**.



- **Apartado 2:** Recorrer el array de artículos y mostrar los datos en formato tabla. Debe resolverse en una función. Los nodos (table, td, tr), deben crearse utilizando los métodos específicos de la estructura DOM.
- **Apartado 3:** Utilizando el mismo identificador del cliente, busca la pizza cuyo identificador sea igual al identificador del cliente. Si lo encuentra muestra los ingredientes de la pizza, creando la estructura de nodos para que tenga el siguiente aspecto.



- **Apartado 4:** Crear botón Comprar:
- **Apartado 5:** Al pulsar el botón Comprar, almacena en el LocalStorage.



- **Apartado 6:** Utilizar find, funciones flecha, o/y map, declaración correcta de variables, etc.
- **Apartado 7:** Al cargar los ingredientes de la pizza que el resultado se muestre transcurrido unos segundos.

- **Valoración**:
  - **Apartado 1: (1 punto)**
  - **Apartado 2: (1 punto)**
  - **Apartado 3: (1 punto)**
  - **Apartado 4: (0,75 puntos)**
  - **Apartado 5: (0,75 puntos)**
  - **Apartado 6: (1 punto)**
  - **Apartado 7: (0,5 puntos)**