## *EXAMEN – 2ª EVALUACIÓN (10 PUNTOS)*

***Nombre:***

Debes entregar los 3 **archivos** en una carpeta con tu nombre : **apellido_nombre**. Los nombres de los ficheros deben ser:

- **apellido_nombre_ejercicio1.html**
- **apellido_nombre_ejercicio2.html**
- **apellido_nombre_ejercicio3.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.

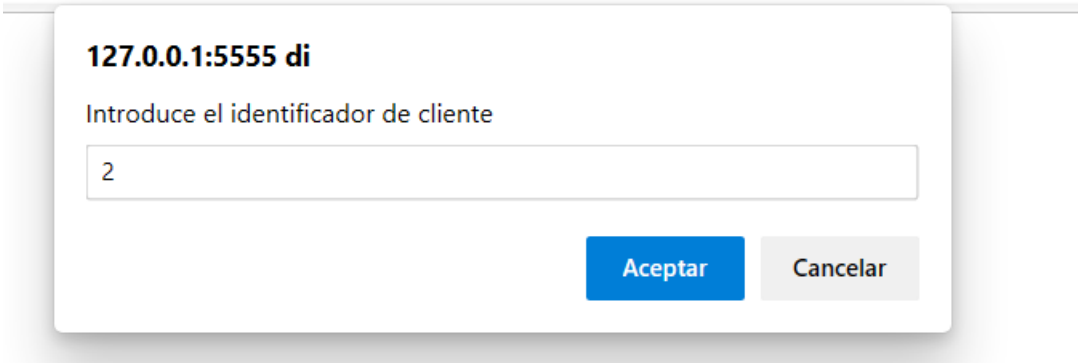
elemento_padre.insertBefore(nuevo_nodo,nodo_de_referencia);

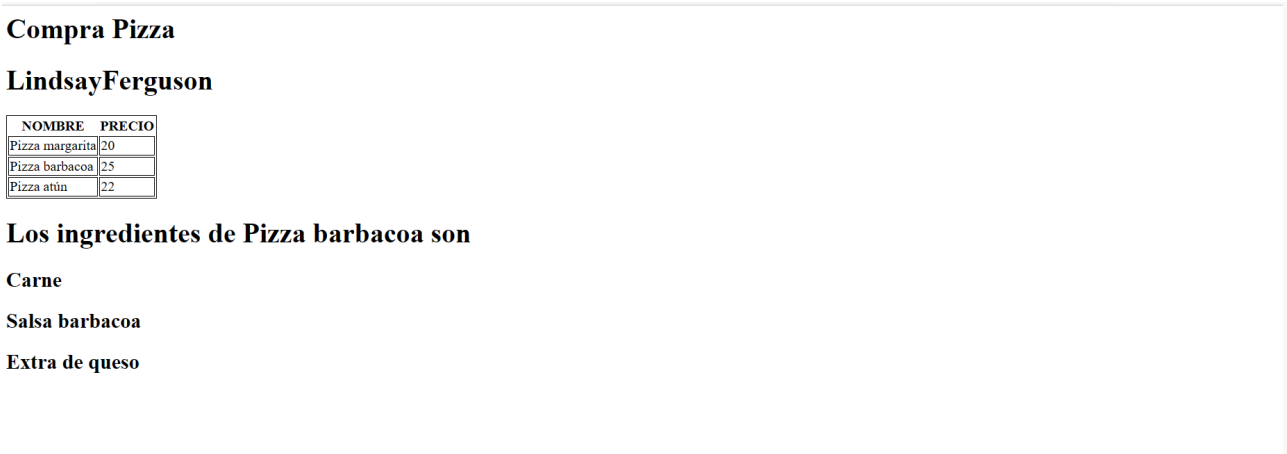
- **insertAfter:** No hay un metodo que inserte un nodo detrás de otro
- **replaceChild:** Para reemplazar un nodo por otro contamos con replaceChild, cuya sintaxis es:
  elemento_padre.replaceChild(nuevo_nodo,nodo_a_reemplazar);
- **removeChild:** Dado que podemos incluir nuevos hijos en un nodo, tiene 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.

1. **Ejercicio 1: Crea un programa JavaScript que se ejecute en el navegador en el cual vamos a trabajar con clases (4 puntos)**
   - **Crea una clase con el nombre de "Vehículo". Esta clase creará objetos de tipo vehículo con las siguientes características:**
     - **Método constructor que permita dar valor inicial a las siguientes propiedades:**
       - **Modelo**
       - **Marca**
       - **Precio**
       - **Km**
     - **Métodos que permitan cambiar (setter) y obtener (getter) información de cada una de las propiedades.**
     - **Método que muestre la marca y el modelo del coche (debe devolver un String)**
- **Debes crear un array de vehículos. Los datos de cada objeto se pueden pedir al usuario o se pueden asignar por código.**
- **Una vez creados los vehículos, deben almacenarse en el almacenamiento local. La clave es el campo Modelo**
- **En la página Web debe aparecer un botón Mostrar que al pulsar recorra todo el LocalStorage y lo visualice. Se valorará mostrarlo con formato**
- **Pedir al usuario un modelo de coche y comprobar si se encuentra en el LocalStorage. Si se localiza muestra sus campos y permite la modificación de cualquiera de sus campos (excepto la clave)**

- **Valoración: La valoración máxima de cada apartado se obtiene en función de la correcta resolución del apartado**
  - **Crear la clase Vehiculo y objetos: (1 punto)**
  - **Almacenar en LocalStorage: (1 punto)**
  - **Mostrar los vehículos: (1 punto)**
  - **Encontrar el coche almacenado y mostrar mensaje (1 punto)**

2. **Ejercicio 2: Crea un programa JavaScript que se ejecute en el navegador que acceda a ficheros json y muestra el siguiente aspecto y funcionalidad (4,5 puntos)**




- **Funcionamiento:**
  - El usuario introduce un identificador de cliente con prompt
  - Acceder a los datos de los usuarios mediante la interfaz Fetch (archivo users.json) y visualizar el nombre del cliente.

- Acceder a los datos de las pizzas mediante la interfaz Fetch (archivo datos.json) y visualizar con formato
- Crear una promesa desde cero en la que se comprueba si existe una pizza con el mismo identificador que el id de cliente. En caso de que la encuentre muestra sus ingredientes
- Se valorará la utilización de distintos tipos de bucles, estructura DOM para crear y visualizar la información

- **Valoración**:
  - **Comprobar que el usuario introduce un identificador (0,5 puntos)**
  - **Fetch users.json y visualizar nombre: (1 punto)**
  - **Fetch datos.json y visualizar pizzas con formato:: (1,50 puntos)**
  - **Crear promesa desde cero y visualizar ingredientes: (2 puntos)**
  - **Código estructurado, con funciones, declaración variables, etc : (0,50 puntos)**

3. **Ejercicio 3: Responde a las preguntas del archivo ejercicio3.js (1,5 puntos)**