

Laboratorio 03: Inicios en JavaScript

Introducción

¿Qué es JavaScript?

JavaScript (JS) es un lenguaje de programación ligero, interpretado, o compilado justo-a-tiempo (just-in-time) con funciones de primera clase. Si bien es más conocido como un lenguaje de scripting (secuencias de comandos) para páginas web, también es usado en muchos entornos fuera del navegador, tal como Node.js. JavaScript es un lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo programación funcional). > MDN

Objetivos

- Desarrollar Habilidades de Programación **m**ejorando la escritura y comprensión de código en JavaScript.
- Fomentar el Pensamiento Lógico, estimulando la resolución de problemas mediante algoritmos.
- Mejorar la Validación de Entradas, mostrando la importancia de validar datos de entrada y manejar errores.

Sintaxis de JS

funciones

Las funciones son uno de los bloques de construcción fundamentales en JavaScript. Una función en JavaScript es similar a un procedimiento — un conjunto de instrucciones que realiza una tarea o calcula un valor, pero para que un procedimiento califique como función, debe tomar alguna entrada y devolver una salida donde hay alguna relación obvia entre la entrada y la salida.


```
function square(number) {  
  return number * number;  
}
```

Arrow functions

son una característica introducida en ECMAScript 6 (también conocido como ES6) que proporciona una sintaxis más concisa y una forma más breve de escribir funciones en JavaScript.

La sintaxis básica de una arrow function se ve así:

```
const sum = (a, b) => a + b;
```

 Para la investigación y mejorar la comprensión, se coloca la siguiente documentación que presenta información sobre las funciones de JS de MDN. Esta fuente es útil para expandir los conocimientos y conocer más sobre las diferentes propiedades, su uso y aplicación en el diseño web.

Métodos de JS recomendados

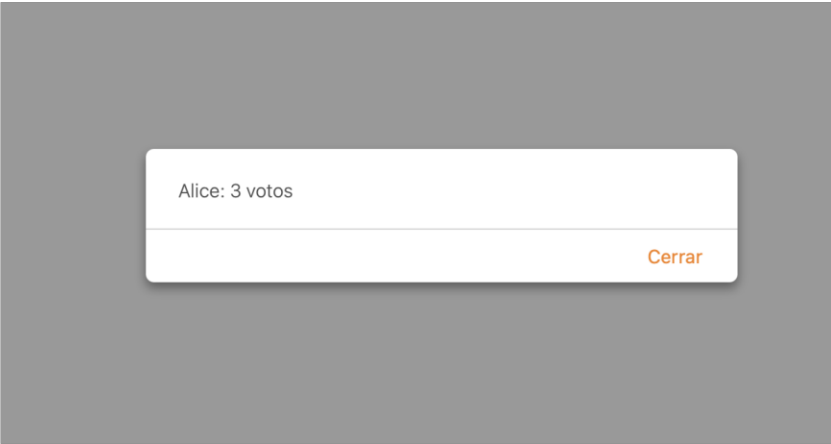
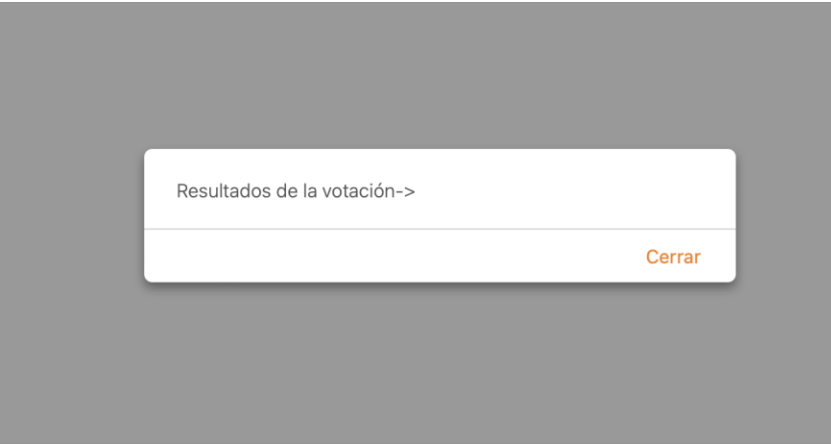
Para obtener más información sobre los métodos presentados, así como otros que pueden ser de utilidad, se recomienda consultar la siguiente documentación en. [MDN](#)

```
trim()  
reverse()  
replace()  
toLowerCase()  
indexOf()
```

```
every()  
length()  
filter()  
slice()  
isNaN()
```

Ejercicio 01

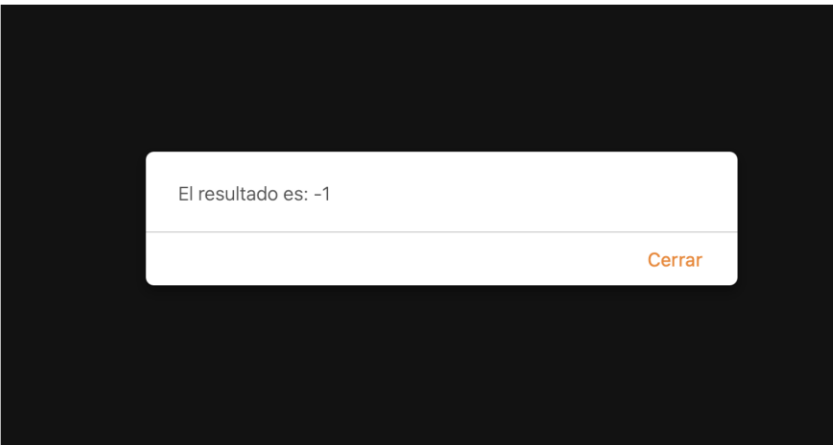
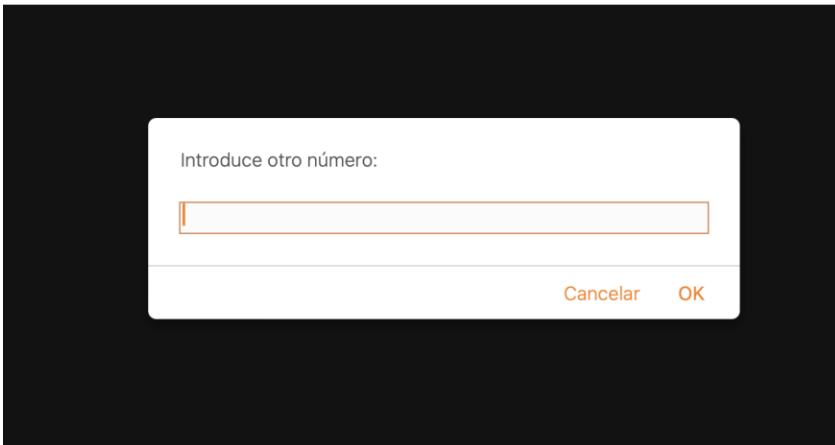
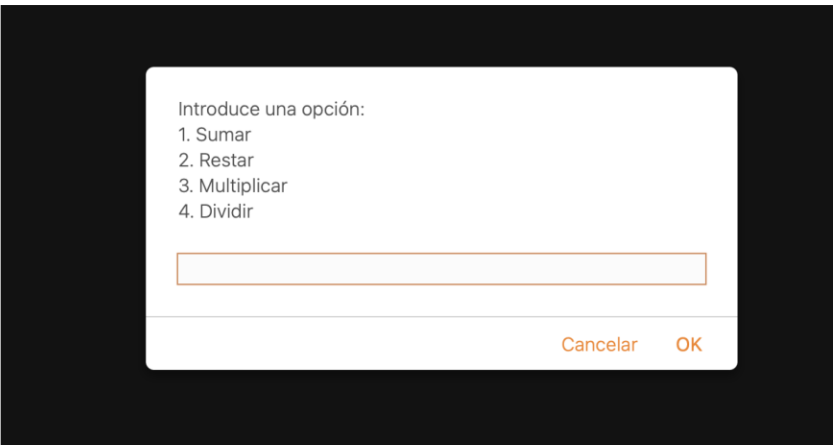
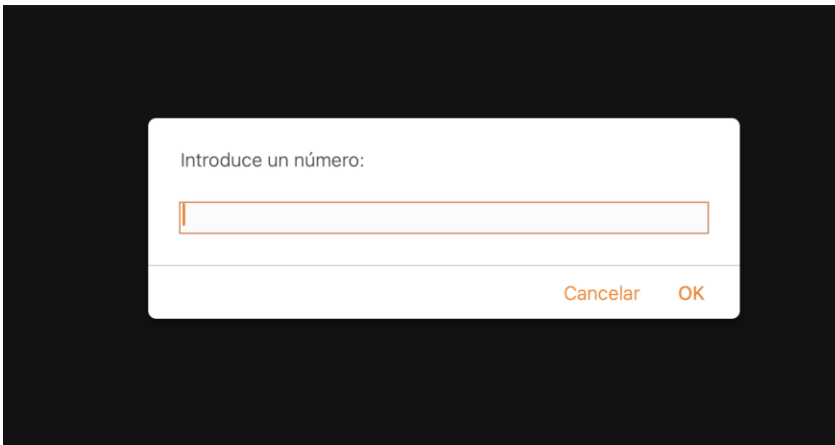
Para esta tarea, se proporcionan tres funciones en JavaScript. Su labor consistirá en integrarlas en un flujo de votación. En primer lugar, se debe definir un array llamado `votes` que simule una votación con varios candidatos, utilizando objetos que contengan la propiedad `candidate`. A continuación, se usará la función `countVotes` para contar cuántos votos ha recibido cada candidato y se almacenará ese resultado en una variable. Finalmente, se utilizará la función `showResults` para mostrar los resultados en un alert. Todo esto se debe ejecutar dentro de una función `main` que coordine el proceso completo.



A continuación, se muestra un ejemplo del resultado esperado para esta sección. La imagen ilustra el diseño final que debe lograrse.

Ejercicio 02

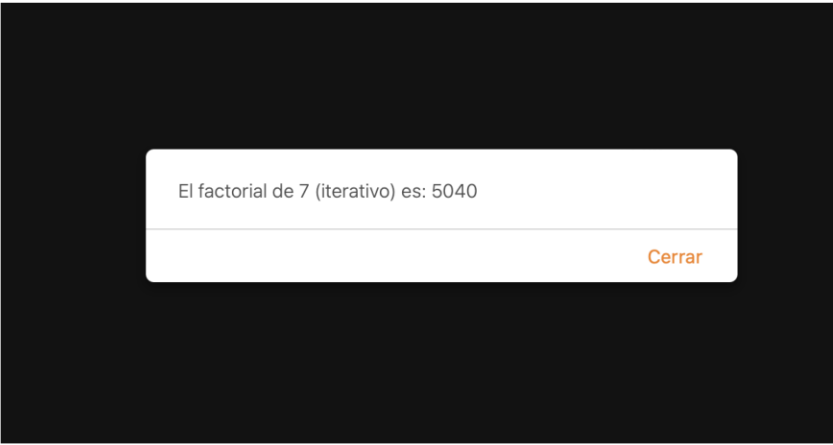
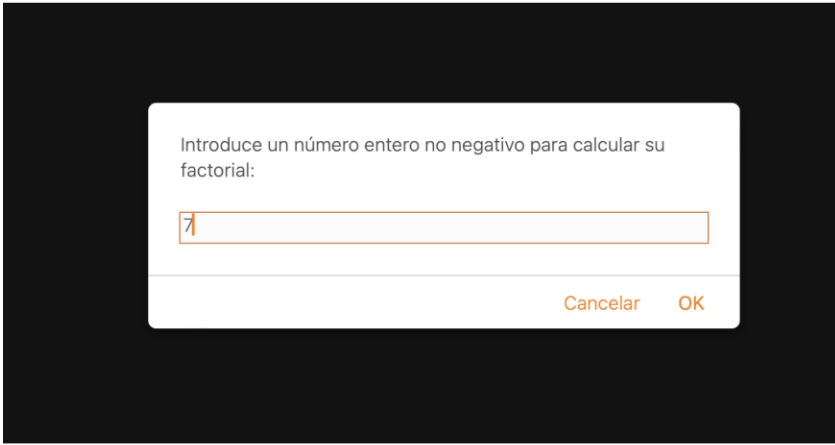
Para esta tarea, se proporcionan varias funciones en JavaScript que permitirán realizar operaciones matemáticas básicas. Primero, se debe utilizar la función `requestNumber` para pedir al usuario que ingrese dos números, validando que la entrada sea un número válido. Luego, se usará la función `selectOperation` para que el usuario elija entre sumar, restar, multiplicar o dividir, asegurándose de manejar el caso en que se intente dividir por cero. Finalmente, se combinarán estos pasos dentro de la función `main`, donde se ejecutará la operación seleccionada con los dos números ingresados y se mostrará el resultado mediante un `alert`.



A continuación, se muestra un ejemplo del resultado esperado para esta sección. La imagen ilustra el diseño final que debe lograrse.

Ejercicio 03

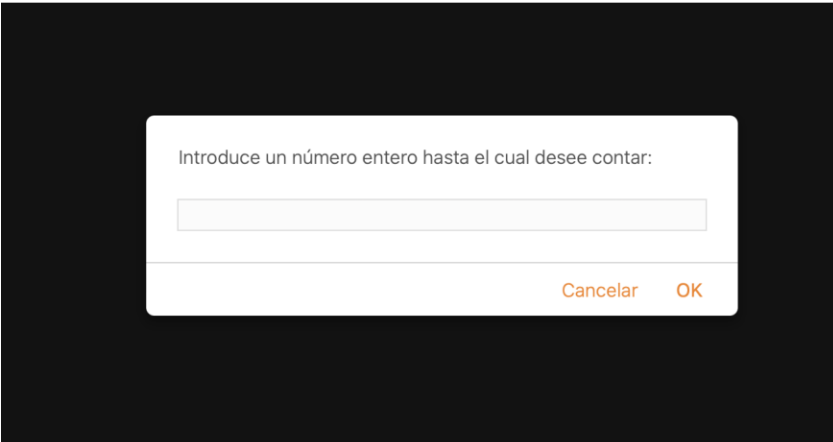
Para esta tarea, se debe calcular el factorial de un número utilizando un enfoque iterativo. Primero, se implementará la función `requestNumber` para pedir al usuario un número entero no negativo, validando que la entrada sea válida. Luego, se usará la función `calculateFactorial` para calcular el factorial de ese número mediante un bucle que multiplica secuencialmente los números hasta el valor ingresado. Finalmente, se integrará todo esto en la función `main`, que solicitará el número, calculará el factorial usando el método iterativo, y mostrará el resultado en una alerta.



A continuación, se muestra un ejemplo del resultado esperado para esta sección. La imagen ilustra el diseño final que debe lograrse.

Ejercicio 04

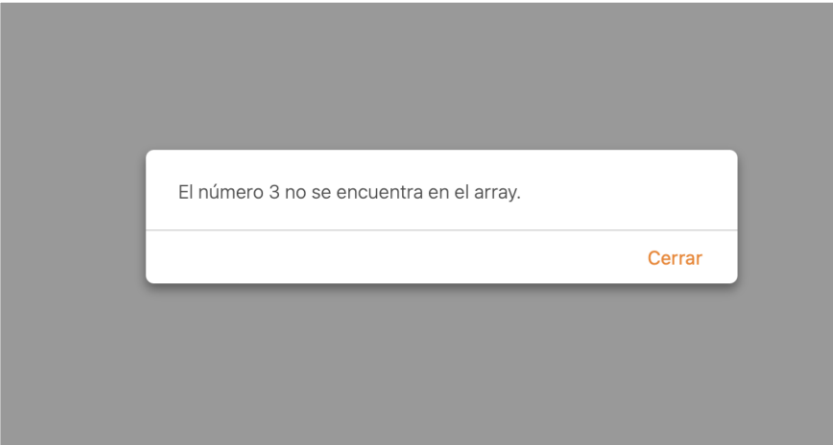
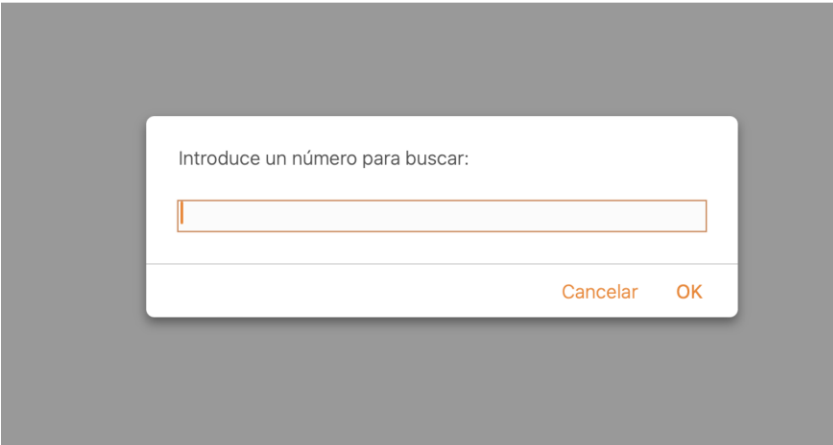
Para esta tarea, se debe crear un contador que incremente su valor cada vez que se le llame. Primero, se implementará la función `createCounter`, que devolverá otra función que incrementa y retorna el valor de un contador interno. Luego, se diseñará la función `requestNumber` para pedir al usuario un número entero no negativo hasta el cual desea contar, validando la entrada. Finalmente, en la función `main`, se llamará a `requestNumber` para obtener el límite de conteo, se creará un contador utilizando `createCounter`, y se usará la función `showCounter` para mostrar el valor del contador en alertas desde 0 hasta el número ingresado.



A continuación, se muestra un ejemplo del resultado esperado para esta sección. La imagen ilustra el diseño final que debe lograrse.

Ejercicio 05

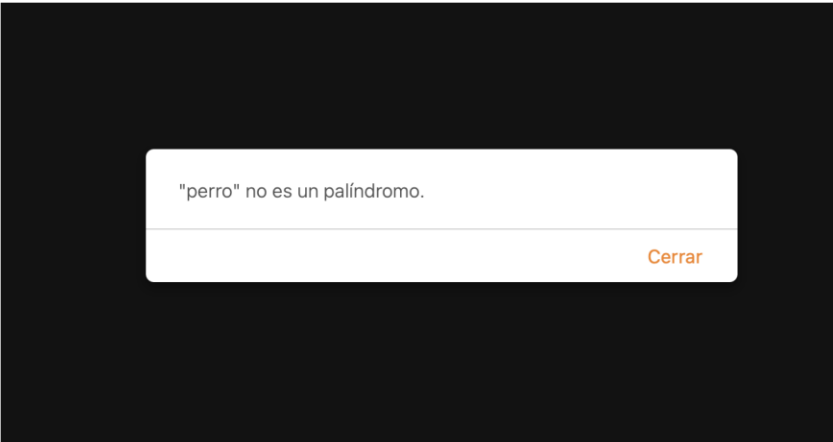
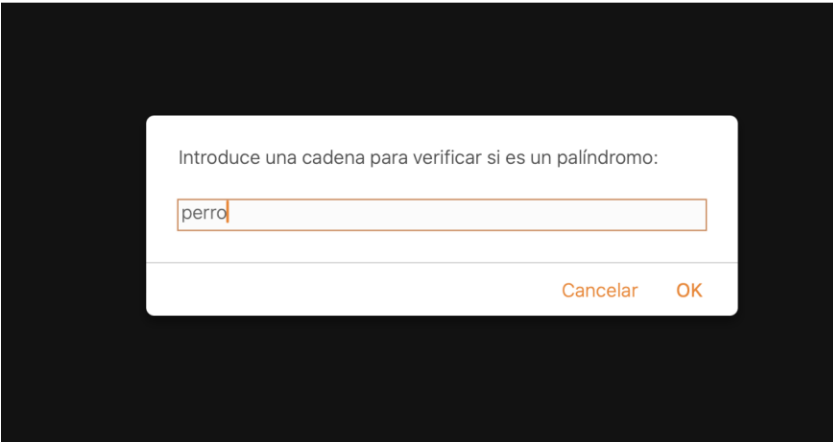
Para esta tarea, se debe implementar un programa que busque un número en un array. Se comenzará con la función `requestNumber`, que solicitará al usuario que ingrese un número, validando que la entrada sea válida. Luego, se utilizará la función `searchNumber` para buscar la posición del número ingresado en un array predefinido de números. En la función `main`, se definirá el array `numbers`, se llamará a `requestNumber` para obtener el número objetivo, y se utilizará `searchNumber` para encontrar su índice. Finalmente, se mostrará un mensaje al usuario indicando si el número se encontró y en qué índice, o si no se encuentra en el array.



A continuación, se muestra un ejemplo del resultado esperado para esta sección. La imagen ilustra el diseño final que debe lograrse.

Ejercicio 06

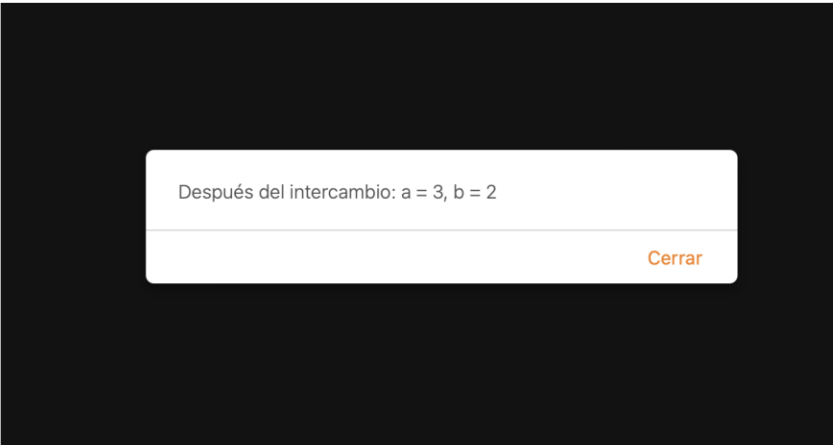
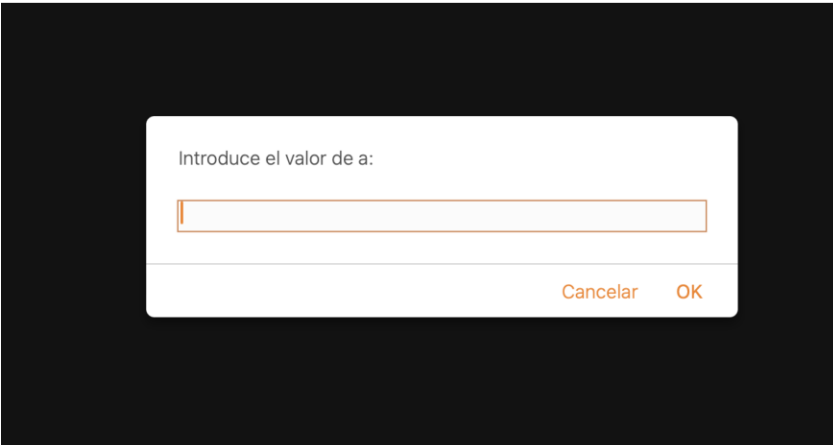
Para esta tarea, se debe implementar un programa que verifique si una cadena es un palíndromo. Se comenzará con la función `requestChain`, que pedirá al usuario que ingrese una cadena, asegurándose de que no esté vacía. Luego, se utilizará la función `isPalindrome` para limpiar la cadena (eliminando caracteres no alfanuméricos y convirtiéndola a minúsculas) y verificar si es igual a su reverso. En la función `main`, se llamará a `requestChain` para obtener el texto a evaluar, y luego se usará `isPalindrome` para determinar si es un palíndromo. Finalmente, se mostrará el resultado al usuario mediante un alert, indicando si la cadena es o no un palíndromo.



A continuación, se muestra un ejemplo del resultado esperado para esta sección. La imagen ilustra el diseño final que debe lograrse.

Ejercicio 07

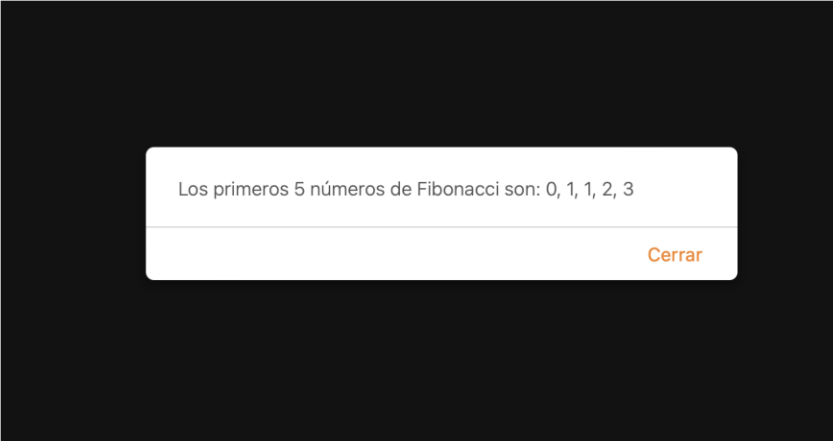
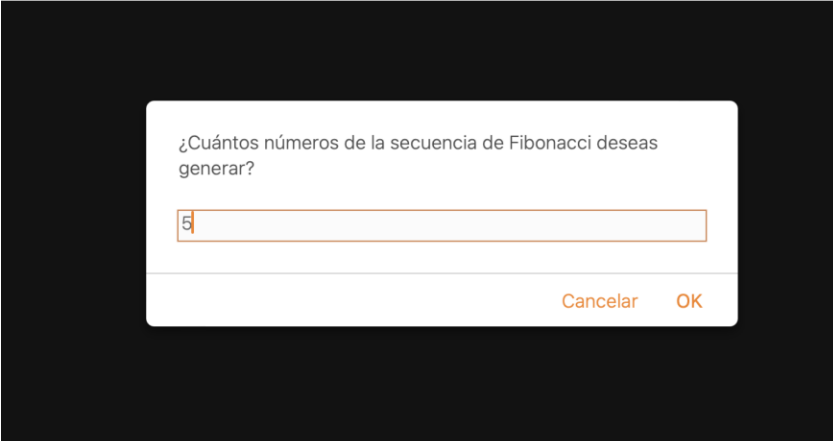
Para esta tarea, se debe crear un programa que solicite al usuario dos valores y los intercambie. Se comenzará implementando la función `requestValue`, que pedirá al usuario que ingrese un valor específico, asegurándose de que no se ingrese una cadena vacía ni que se cancele el prompt. Luego, en la función `main`, se llamará a `requestValue` dos veces para obtener los valores `a` y `b`. Se utilizará la desestructuración de arrays para intercambiar los valores de `a` y `b`. Finalmente, se mostrará un mensaje al usuario con el resultado del intercambio mediante un alert, indicando los nuevos valores de `a` y `b`.



A continuación, se muestra un ejemplo del resultado esperado para esta sección. La imagen ilustra el diseño final que debe lograrse.

Ejercicio 08

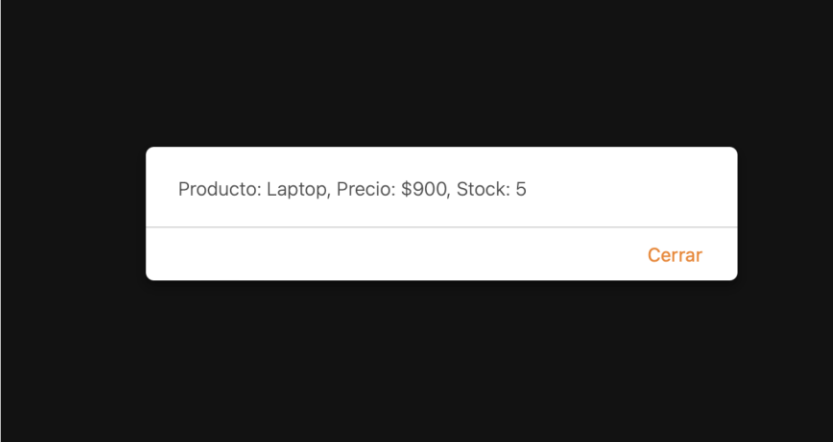
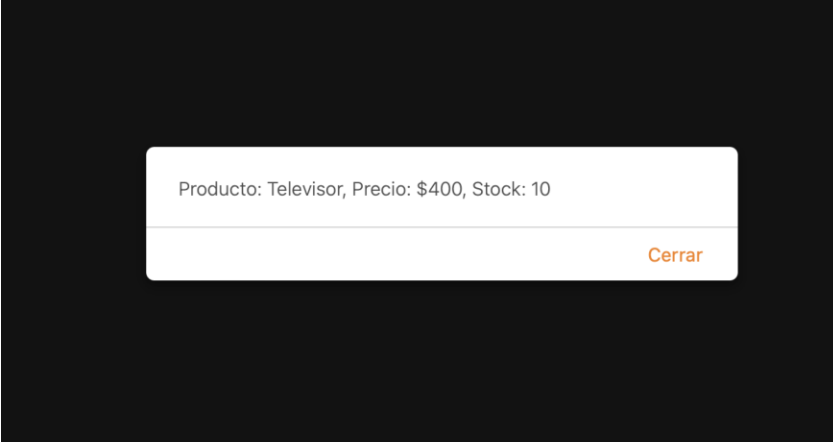
Para esta tarea, se debe implementar un programa que genere la secuencia de Fibonacci hasta un número específico de elementos. Se comenzará con la función `requestQuantity`, que pedirá al usuario cuántos números de la secuencia desea generar, validando que la entrada sea un número entero positivo. Luego, se utilizará la función `fibonacciCalculate` para calcular la secuencia, que comienza con los números 0 y 1, y suma los dos anteriores para obtener los siguientes elementos. En la función `main`, se llamará a `requestQuantity` para obtener la cantidad deseada de números, y luego se usará `fibonacciCalculate` para generar la secuencia. Finalmente, se mostrará un mensaje al usuario con los números generados mediante un alert, y se manejará el caso donde la cantidad solicitada sea cero, indicando que no se generaron números de Fibonacci.



A continuación, se muestra un ejemplo del resultado esperado para esta sección. La imagen ilustra el diseño final que debe lograrse.

Ejercicio 09

Para esta tarea, se debe implementar un programa que filtre y muestre productos disponibles en stock. Se comenzará con la función `filterAvailableProducts`, que recibe un array de productos y retorna solo aquellos que tienen stock mayor a cero. Luego, se implementará la función `showProducts`, que mostrará los detalles de cada producto disponible mediante alertas; si no hay productos en stock, deberá informar al usuario que no hay productos disponibles. En la función `main`, se definirá un array de productos con sus nombres, precios y niveles de stock. Se llamará a `filterAvailableProducts` para obtener solo los productos disponibles y luego se usará `showProducts` para presentar los resultados al usuario.



A continuación, se muestra un ejemplo del resultado esperado para esta sección. La imagen ilustra el diseño final que debe lograrse.

Ejercicio 10

Para esta tarea, se debe implementar un programa que filtre y muestre a los estudiantes aprobados según sus notas. Se comenzará con la función `filterApproved`, que toma un array de estudiantes y devuelve solo aquellos cuyos todas sus notas son mayores o iguales a 5, utilizando el método `every`. Luego, se implementará la función `showApproved`, que mostrará un mensaje alertando los nombres de los estudiantes aprobados; si no hay aprobados, deberá informar al usuario que no hay estudiantes aprobados. En la función `main`, se definirá un array de estudiantes con sus nombres y notas. Se llamará a `filterApproved` para obtener la lista de estudiantes aprobados y luego se usará `showApproved` para presentar los resultados al usuario.

