Promesas o promises

En esta sección, se realizará una simulación de un retraso en una tarea, esto va a proporcionar una idea sobre lo que es una promesa en JavaScript

Simulando un delay en una tarea asíncrona

Las promesas en JavaScript se utilizan para manejar tareas asíncronas, es decir, tareas que pueden tomar algún tiempo para completarse. Las promesas permiten realizar una tarea y, en lugar de esperar su finalización, se continúa con otras tareas.

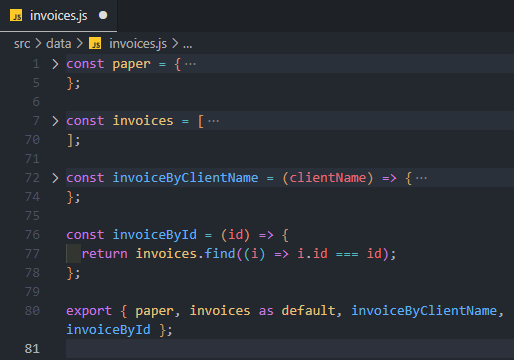
En el contexto del frontend, esto puede referirse a la obtención de datos de una API remota a través del backend. Un "delay" se refiere a un retraso; no hay garantía del 100% de que la promesa se cumpla correctamente.

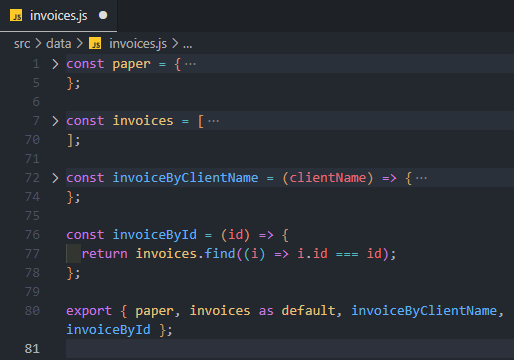
Sigue estos pasos para realizar la simulación del retraso:

Paso 1: Definir una función exportable en el archivo invoices

En esta sección se van a utilizar los archivos **main.js** (debe estar en blanco) e **invoices.js** (el mismo archivo de la sección anterior).

En el archivo **invoices.js**, crea una función exportable para buscar por **id** con un parámetro, pues esta función se va a utilizar en una promesa con JavaScript.



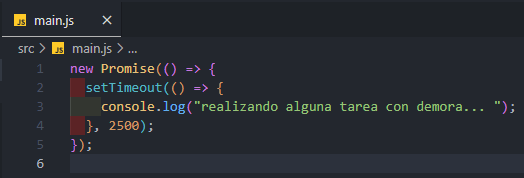


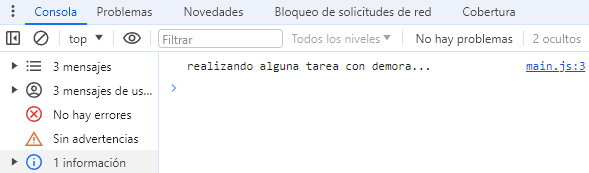
Paso 2: Simular una promesa en el archivo main

Es importante recordar que, aunque todo el código es secuencial y síncrono cuando se ejecuta un archivo JavaScript, las promesas son asíncronas y se ejecutan en un espacio distinto a la ejecución normal. Primero se ejecuta todo lo síncrono (todo el código lineal), y luego lo asíncrono.

El constructor **Promise** se utiliza principalmente para envolver funciones que no admiten promesas de forma nativa, especialmente las funciones que operan de manera asíncrona utilizando callbacks.

El método **setTimeout** se utiliza para simular un delay. Este método recibe dos argumentos: un callback o una función de flecha que se ejecutará después de un cierto tiempo y el segundo argumento es el tiempo en milisegundos; en este caso, la función se va a ejecutar después de **2500** milisegundos.

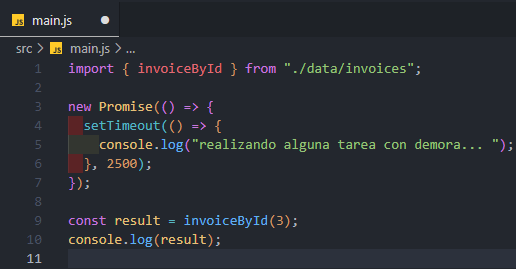


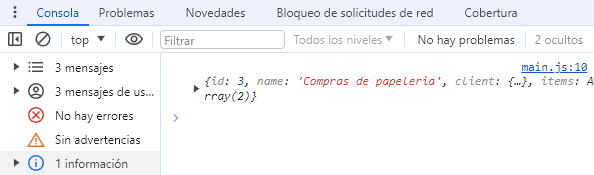


Paso 3: Importar la función definida en el archivo main

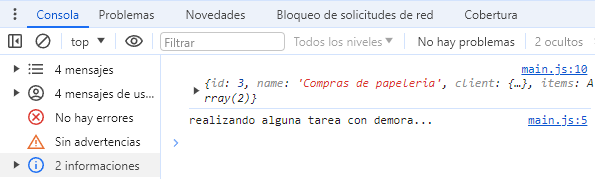
Ten en cuenta que primero se ejecuta la tarea síncrona, que es el código que se encuentra fuera de la promesa, y luego se ejecuta la tarea asíncrona que se simula con la promesa.

En el siguiente ejemplo, en la consola, primero se verá el resultado de la tarea síncrona (el llamado a la función **invoiceById**) y luego el mensaje de la tarea asíncrona después del retraso.





Luego de 2,5 segundos se muestra el mensaje en la consola:



Paso 4: Simular una petición al backend

Coloca la función **invoiceById** dentro del método **setTimeout** para simular una petición a un API remoto. En este caso, se va a consumir un servicio en particular para obtener los datos de una factura por su ID.

