API DOM (Document Object Model)

En esta sección, vamos a mostrar los nombres de los usuarios en una página web en lugar de la consola manipulando el DOM.

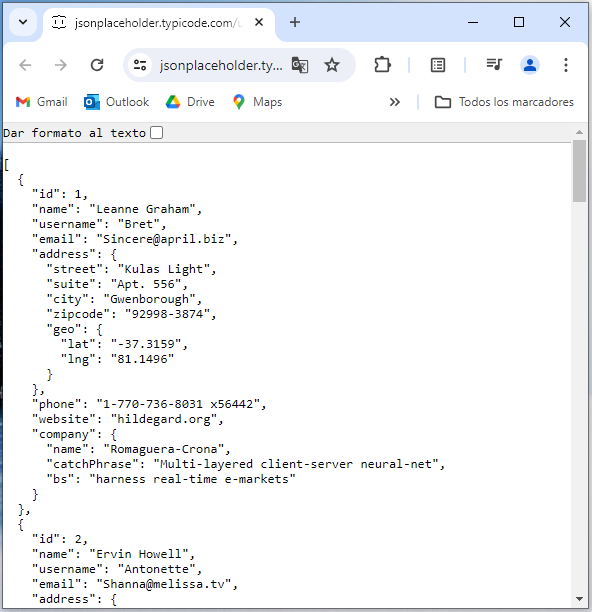
Mostrar el nombre de cada uno de los usuarios

Para mostrar el nombre de cada usuario en el documento HTML en lugar de en la consola, es necesario manipular el DOM (Document Object Model), que es básicamente una interfaz de programación para documentos HTML.

Paso 1: Consumir el API y mostrar los nombres de los usuarios en la consola

Primero, definimos una función asincrona **findAllUsers** para consumir el API y mostrar los nombres de los usuarios en la consola (similar al mismo código de la sección anterior).

Recuerda que al acceder a la URL: <https://jsonplaceholder.typicode.com/users>, nos muestra un codigo JSON. Se puede notar que cada uno de los usuarios tienen definido su nombre en el atributo **name** en cada uno de los objetos contenidos en el arreglo principal.



const findAllUsers = async () => {
  const response = await fetch("https://jsonplaceholder.typicode.com/users");
  const users = await response.json();

  users.forEach((user) => {
    console.log(user.name);
  });
};

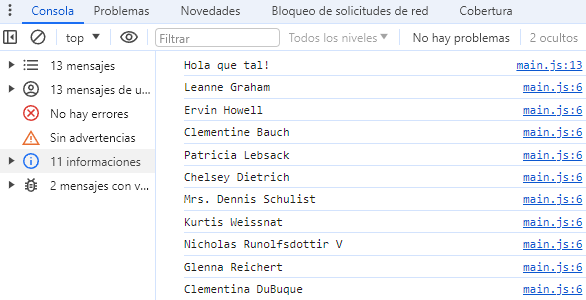
findAllUsers();

// console.log(users);
console.log("Hola que tal!");


Explicación de cada línea de código

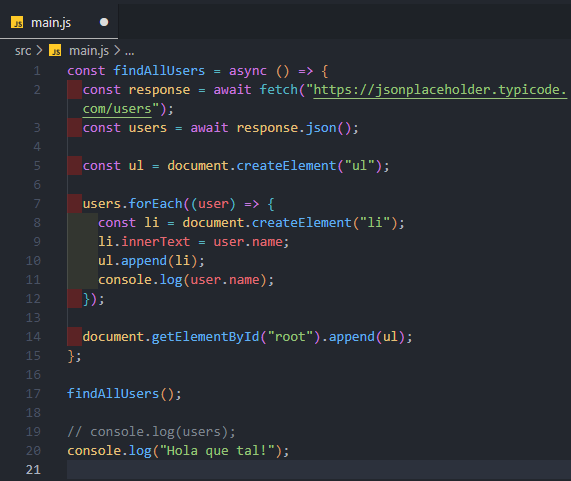
* **const findAllUsers = async () => { ... }:** Esta línea define una función llamada **findAllUsers** que es asincrónica (**async**). Esto significa que la función retornará una promesa y puede usar **await** dentro de ella para esperar que otras promesas se resuelvan.
* **const response = await fetch("https://jsonplaceholder.typicode.com/users"):** Aquí, **fetch** se utiliza para hacer una solicitud HTTP GET a la URL proporcionada, que devuelve una promesa que se resuelve en un objeto Response. Con **await**, esperamos a que esta promesa se resuelva y almacenamos el resultado en la variable **response**.
* **const users = await response.json():** Se utiliza **await** nuevamente para esperar a que la promesa devuelta por **response.json** se resuelva. Esta función convierte el cuerpo de la respuesta en JSON y devuelve otra promesa que se resuelve en los datos JSON. Estos datos JSON representan los usuarios obtenidos de la API.
* **users.forEach((user) => { console.log(user.name) }):** Se utiliza **forEach** para iterar sobre el arreglo de usuarios (**users**) y para cada usuario, imprimimos su nombre en la consola.

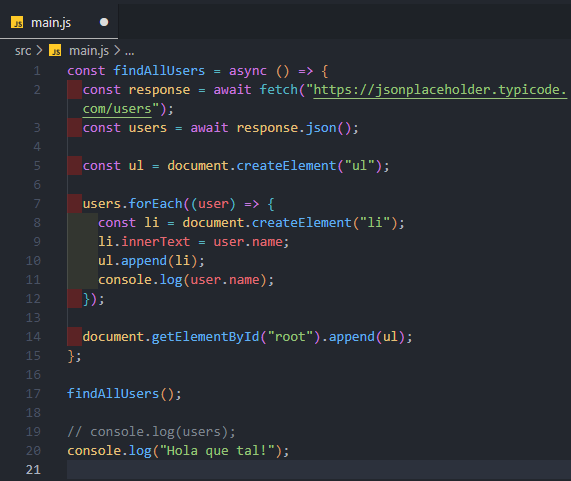
Finalmente, la función **findAllUsers** se llama, lo que desencadena la ejecución de todo el código dentro de ella. Luego, se imprime el mensaje **"Hola que tal!"** en la consola. Como resultado, primero se muestra el mensaje y luego se imprimen los nombres de los usuarios.



Paso 2: Mostrar los nombres en una lista HTML

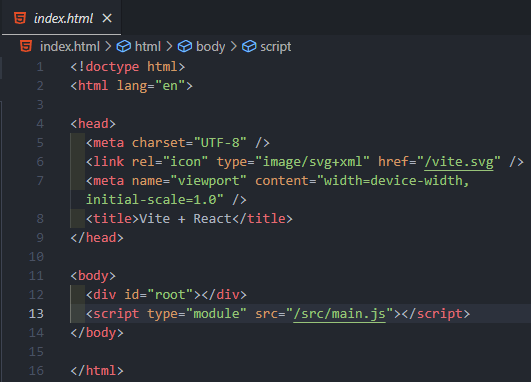
Para mostrar los nombres en el documento HTML en una lista, primero creamos un elemento **<ul>** y luego, por cada usuario, creamos un elemento **<li>** y lo agregamos al **<ul>**. Todo esto se puede realizar manipulando el DOM.

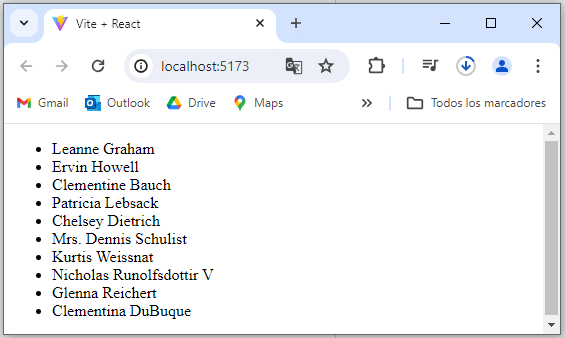


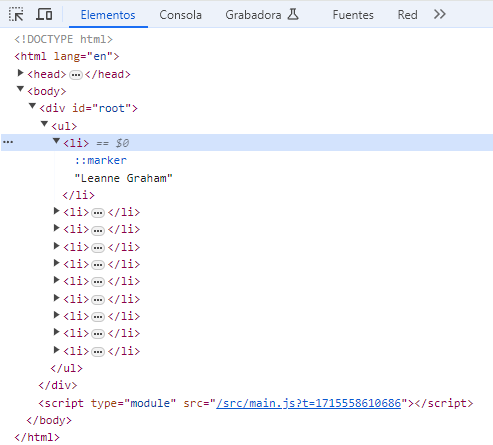


Explicación de cada línea de código

* **const ul = document.createElement("ul"):** Crea un nuevo elemento **<ul>** que contendrá la lista de nombres de los usuarios.
* **users.forEach((user) => { ... }):** Utiliza el método **forEach** para iterar sobre la lista de usuarios.
* **const li = document.createElement("li"):** Crea un nuevo elemento **<li>** para cada usuario.
* **li.innerText = user.name:** Establece el texto interno del **<li>** con el nombre del usuario.
* **ul.append(li):** Agrega cada **<li>** al **<ul>**.
* **document.getElementById("root").append(ul):** Utiliza el método **append** para agregar el elemento **<ul>** completo al elemento con el ID **"root"** en el documento HTML. Este último elemento debe existir en el archivo **index.html**.







El resultado es una lista de nombres de usuarios que se muestra tanto en la consola como en el documento HTML.

Lo que hemos hecho aquí manualmente es el tipo de manipulación del DOM que React automatiza para hacer más sencillo y eficiente el desarrollo web. React es una biblioteca de JavaScript que simplifica la creación de interfaces de usuario y el manejo del DOM, permitiendo la creación de componentes reutilizables y una programación más declarativa.