



REPORTE: INTRODUCCIÓN A REACT NATIVE

ESTUDIANTE: PÉREZ MELLADO JENNYFER

GRUPO: 211

MAESTRO: IVAN ISAY GUERRA LOPEZ

MATERIA: DESARROLLO DE APPS MÓVILES

FECHA: 04 - 09 - 2025

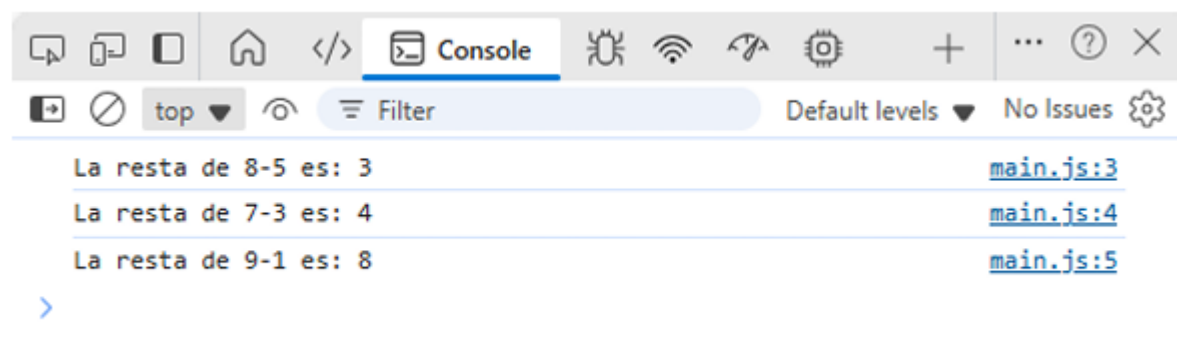
En esta práctica, resolvimos tres ejercicios con el uso de import, export, promises, async y await.

Ejercicio 1:

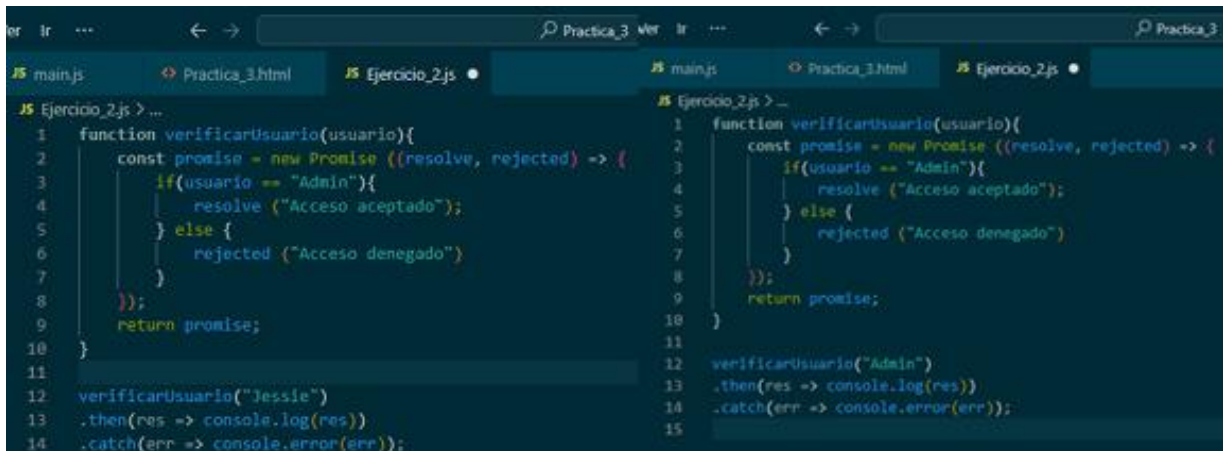
```
JS main.js
1  import {restar} from './utils.js';
2
3  console.log("La resta de 8-5 es: " + restar(8, 5));
4  console.log("La resta de 7-3 es: " + restar(7, 3));
5  console.log("La resta de 9-1 es: " + restar(9, 1));
```

```
JS utils.js > restar
1  export function restar (a, b)
2  {
3    |    return (a - b);
4  }
```

Para el ejercicio, cree dos archivos “.js”, en el primero (segunda imagen), tenemos una función restar la cual cuenta con dos variables como parámetros, la cual resta estos mismos y regrese el resultado de dicha resta, esta misma función se exporta mediante la palabra reservada “export”, en el segundo archivo, primero importamos la función que creamos en el primer archivo, esto mediante la palabra reservada “import”, se pone entre llaves el nombre de la función que se va a utilizar de otro archivo, y después de un “from” la ubicación en donde se encuentra dicha función, ya que ambos archivos se encuentran en la misma carpeta, basta con poner el nombre del archivo y poco más, después solo falta un “console.log” para llamar a la función con ciertas variables y mostrar el resultado en la consola.



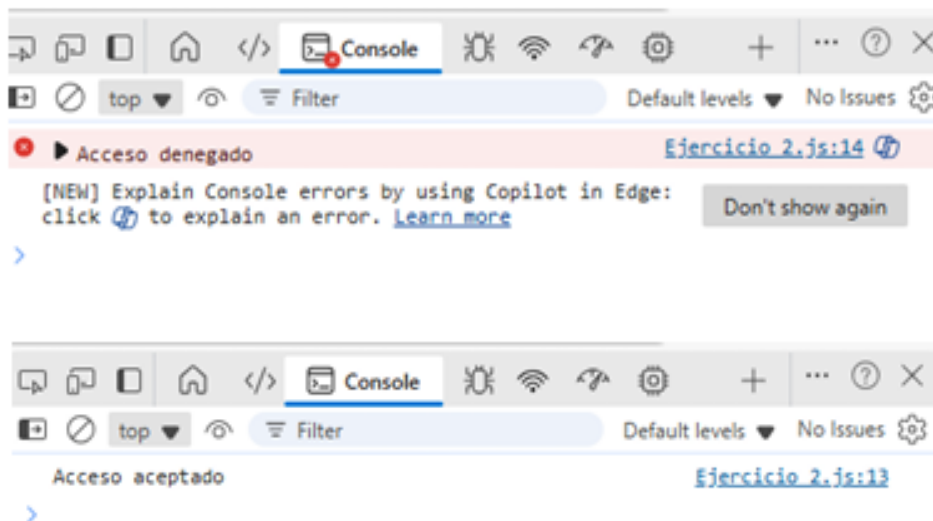
Ejercicio 2:



```
1 function verificarUsuario(usuario){
2   const promise = new Promise ((resolve, rejected) => {
3     if(usuario == "Admin"){
4       resolve ("Acceso aceptado");
5     } else {
6       rejected ("Acceso denegado")
7     }
8   });
9   return promise;
10 }
11
12 verificarUsuario("Jessie")
13 .then(res => console.log(res))
14 .catch(err => console.error(err));
```

```
1 function verificarUsuario(usuario){
2   const promise = new Promise ((resolve, rejected) => {
3     if(usuario == "Admin"){
4       resolve ("Acceso aceptado");
5     } else {
6       rejected ("Acceso denegado")
7     }
8   });
9   return promise;
10 }
11
12 verificarUsuario("Admin")
13 .then(res => console.log(res))
14 .catch(err => console.error(err));
```

En este caso, tenemos una función cuyo objetivo es verificar si se está hablando de un admin o no, por lo que hacemos uso de las promesas para mandar un mensaje de aceptación en caso de ser admin, o un mensaje de rechazo en caso de no serlo; para esto, realice una promesa, tal cual su sintaxis, también con una función flecha agregue un if-else, con una comprobación de si la variable usuario cumple con ser “admin”, por lo que si es correcto, se cumple el resolve y se manda un mensaje de aceptación, de lo contrario con el rejected se envíe el mensaje de denegado; después de esto se llama a la función tal cual se llama y se le entrega su parámetro, ya sea admin para ser aceptado, u otro nombre para ser rechazado, esto seguido del .then para marcar la promesa como correcta o .catch para marcarla como error.



Ejercicio 3:

```
JS Ejercicio_3.js > obtenerDatos > [0] resultado
1  function simularPeticionAPI(){
2      return new Promise(resolve => {
3          |   setTimeout(() => {
4              |       resolve("Datos recibidos correctamente");
5          |   }, 5000);
6      });
7  }
8
9  async function obtenerDatos() {
10     const resultado = await simularPeticionAPI();
11     console.log(resultado);
12 }
13
14 obtenerDatos();
```

Para el tercer ejercicio, tenemos una función async y await, o bueno, tenemos que armar esta misma, la función general ya la tenemos, por lo que el trabajo consiste en usar la función await que llame a la promesa, y esto lo asignamos a una variable constante, para en esta misma función imprimirla usando esta misma variable, de esta forma solo obteniendo el resultado que será la resolución con el texto que tiene, por ultimo solo hacemos el llamado a la función para que esta se ejecute y en la consola podemos ver el resultado:



Conclusión: En esta práctica, pude poner en práctica muchos de los temas que ya hemos vistos, y al menos yo considero que al aplicarlos a ejercicios, puedo entenderlo mejor para su uso futuro, a diferencia de solo verlos en la teoría, tengo la idea y los fundamentos teóricos, pero no funcionales, por lo que al resolver estos ejercicios puedo reflexionar sobre lo que se y buscar la forma de resolver estos ejercicios con estos conocimientos.

