

## Laboratorio #3

**Nombre: Victoria Jimenez Martinez**

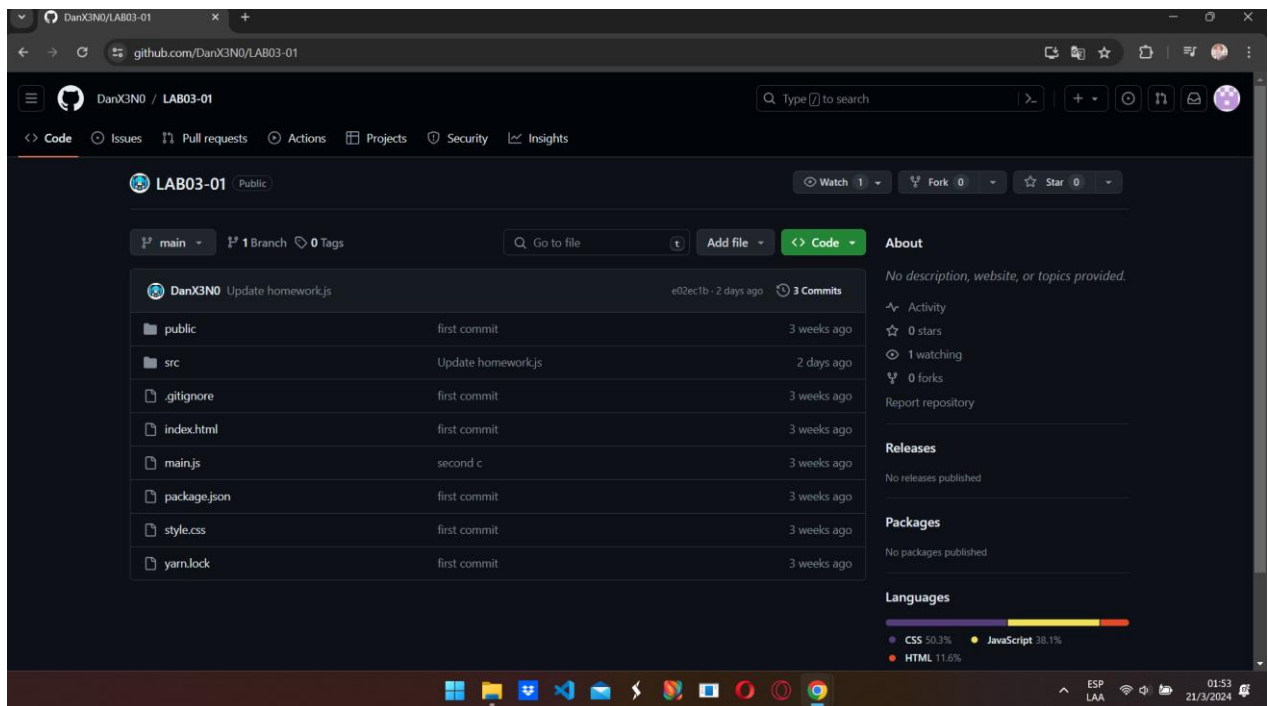
**C.I.: 8610323      R.U.: 103334      Fecha: 21/03/2024**

## GIT Y GITHUB 2 (L - 3)

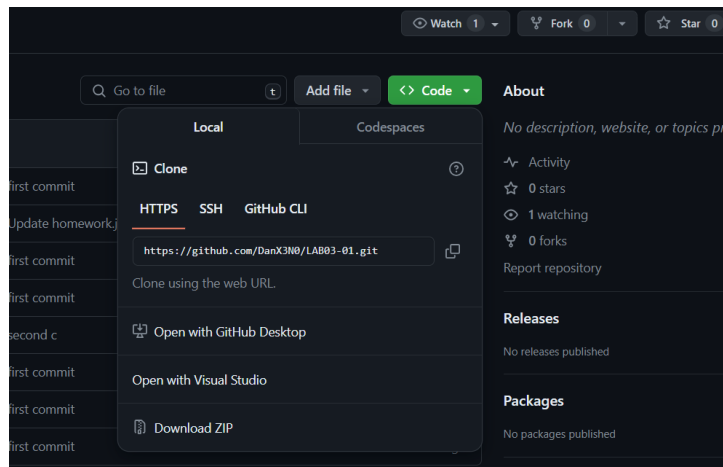
Con lo anterior visto, es hora de dar un paso más allá en lo que respecta al uso de Git. Con este laboratorio tocaremos algunos tópicos interesantes, como lo es el clonning de repos y otros.

### Sobre Git clone

Ingresa al siguiente enlace: <https://github.com/DanX3N0/LAB03-01>



Deberá aparecer algo similar, luego se deberá dar click en Code y mostrará lo siguiente:



Abran la terminal y estando en el directorio donde ustedes quieren clonar el repo, copiar el enlace del repo, y en la terminal insertar el siguiente código:

```
C:\Users\Hp>cd desktop
C:\Users\Hp\Desktop>cd Seminario
C:\Users\Hp\Desktop\Seminario>git clone https://github.com/DanX3N0/LAB03-01.git
Cloning into 'LAB03-01'...
remote: Enumerating objects: 21, done.
remote: Counting objects: 100% (21/21), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 21 (delta 4), reused 15 (delta 2), pack-reused 0
Receiving objects: 100% (21/21), 9.96 KiB | 4.98 MiB/s, done.
Resolving deltas: 100% (4/4), done.
C:\Users\Hp\Desktop\Seminario>
```

Una vez ejecutado el comando, ingresar al proyecto clonado con VS Code o el editor de su preferencia además de instalar las dependencias con el comando “yarn” en la raíz del proyecto. Se debería poder observar la siguiente estructura:

```
C:\Users\Hp\Desktop\Seminario\LAB03-01>yarn
yarn install v1.22.22
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
Done in 32.06s.
C:\Users\Hp\Desktop\Seminario\LAB03-01>
```

## EVALUACIÓN

1.- Una vez ya revisados los archivos, espero hayas comprendido de buena manera su funcionamiento, pero para corroborar eso ahí van algunas preguntas. Dentro de src crea un archivo P-01.txt y explica con un ejemplo cómo funciona el archivo .gitignore, package.json y otro archivo sobre algún archivo o tema que hayas investigado o te haya interesado.

```
C:\Users\Hp\Desktop\Seminario\LAB03-01>cd src
C:\Users\Hp\Desktop\Seminario\LAB03-01\src>echo > P-01.txt
C:\Users\Hp\Desktop\Seminario\LAB03-01\src>
```

The screenshot shows a VS Code editor with a file explorer on the left. The file explorer shows a project structure with folders like 'node\_modules', 'public', and 'src'. The 'src' folder contains files like 'hello.js', 'homework.js', 'P-01.txt', '.gitignore', 'index.html', 'main.js', 'package.json', 'style.css', and 'yarn.lock'. The main editor displays the content of 'P-01.txt', which contains a 'package.json' file and a 'Dockerfile' example. The 'package.json' file is a JSON object with fields for 'name', 'private', 'version', 'type', 'scripts', and 'devDependencies'. The 'Dockerfile' example is a text file with instructions on how to build a Docker image from a Node.js base image.

```
----- package.json: -----
Es un archivo de configuración utilizado en proyectos Node.js. Contiene metadatos del proyecto, como nombre, versión, descripción
EJEMPLO:
{
  "name": "lab03", // Nombre del Proyecto
  "private": true, // Indica que el Proyecto es privado y no será publicado en el registro de paquetes npm.
  "version": "0.0.0", // Versión inicial del Proyecto.
  "type": "module", // Tipo de Módulo que el Proyecto utiliza (En este caso, módulos ES).
  "scripts": {
    "dev": "vite", // Comando para ejecutar el entorno de desarrollo utilizando vite.
    "build": "vite build", // Comando para compilar el Proyecto utilizado Vite.
    "preview": "vite preview" // Comando para previsualizar el proyecto compilado utilizando Vite.
  },
  "devDependencies": {
    "vite": "^5.1.4" // Dependencia de desarrollo del paquete Vite y su versión específica
  }
}

----- Archivo de configuración de Docker (Dockerfile): -----
Es un archivo de texto que contiene los comandos necesarios para construir una imagen de Docker. Estos comandos especifican las c
EJEMPLO:
# Usar la imagen base de Node.js
FROM node:14
```

2.- En el archivo homework.js, resolver los ejercicios propuestos, la solución deberá de estar en ese archivo debajo de la especificación de cada uno, dentro del área delimitada con comentarios. Tal vez te resulte útil investigar un poco acerca de los métodos que se tienen para los arreglos, como ser: map(), filter(), sobre objetos y también deberás pensar en cómo usar arreglos o alguna otra estructura de datos, para resolver el ejercicio # 4.

The screenshot shows a VS Code editor with a file explorer on the left. The file explorer shows a project structure with folders like 'node\_modules', 'public', and 'src'. The 'src' folder contains files like 'hello.js', 'homework.js', 'P-01.txt', '.gitignore', 'index.html', 'main.js', 'package.json', 'style.css', and 'yarn.lock'. The main editor displays the content of 'homework.js', which contains JavaScript code for three exercises. The first exercise involves creating an array of numbers and using the map() method to multiply each element by 5. The second exercise involves creating an array of names and using the sort() method to sort them alphabetically. The third exercise is partially visible.

```
1  (() => {
2
3  /* Exercices */
4
5  /* First exercise */
6
7  let numbers : number[] = [1, 2, 3, 4, 5];
8
9
10 /* Multiplique los numeros del arreglo por 5 */
11 let multipliedNumbers : number[] = numbers.map(number => number * 5);
12
13 /*Resultado esperado: [5, 10, 15, 20, 25] */
14 console.log("Resultado del primer ejercicio: ", multipliedNumbers);
15
16 /* First exercise */
17
18 /* Second exercise */
19
20 let names : string[] = ["Jhosep", "Daniel", "Rodrigo", "Veronica"];
21
22 /* Ordenar los nombres alfabeticamente */
23 let sortedNames : string[] = names.sort();
24
25 /* Resultado esperado: ["Daniel", "Jhosep", "Rodrigo", "Veronica"] */
26 console.log("Resultado del segundo ejercicio: ", sortedNames);
27
28 /* Second exercise */
29
30 /* Thrid exercise */
```

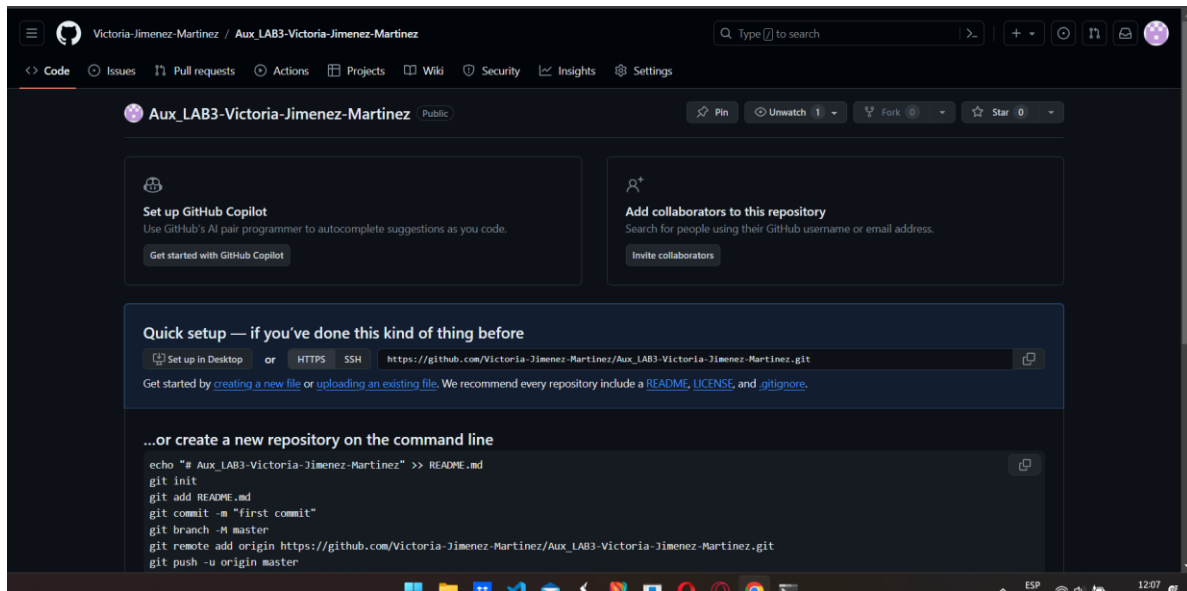
```
C:\Users\Hp\Desktop\Seminario\LAB03-01\src>node homework.js
Resultado del primer ejercicio: [ 5, 10, 15, 20, 25 ]
Resultado del segundo ejercicio: [ 'Daniel', 'Jhosep', 'Rodrigo', 'Veronica' ]
Resultado del tercer ejercicio: [ 'a', 'b', 'z', 'r', 'c' ]
Resultado del cuarto ejercicio: { a: 2, b: 2, c: 2, d: 2, A: 1, B: 1, C: 1, D: 1 }

C:\Users\Hp\Desktop\Seminario\LAB03-01\src>node homework.js
Resultado del primer ejercicio: [ 5, 10, 15, 20, 25 ]
Resultado del segundo ejercicio: [ 'Daniel', 'Jhosep', 'Rodrigo', 'Veronica' ]
Resultado del tercer ejercicio: [ 'a', 'b', 'z', 'r', 'c' ]
Resultado del cuarto ejercicio: { a: 2, b: 2, c: 2, d: 2, A: 1, B: 1, C: 1, D: 2 }

C:\Users\Hp\Desktop\Seminario\LAB03-01\src>node homework.js
Resultado del primer ejercicio: [ 15, 30, 45, 60, 75 ]
Resultado del segundo ejercicio: [ 'Daniel', 'Jhosep', 'Rodrigo', 'Veronica' ]
Resultado del tercer ejercicio: [ 'a', 'b', 'z', 'r', 'c' ]
Resultado del cuarto ejercicio: { a: 2, b: 2, c: 2, d: 2, A: 1, B: 1, C: 1, D: 2 }

C:\Users\Hp\Desktop\Seminario\LAB03-01\src>node homework.js
Resultado del primer ejercicio: [ 5, 10, 15, 20, 25 ]
Resultado del segundo ejercicio: [ 'Daniel', 'Jhosep', 'Rodrigo', 'Veronica' ]
Resultado del tercer ejercicio: [ 'a', 'b', 'z', 'r', 'c' ]
Resultado del cuarto ejercicio: { a: 2, b: 2, c: 2, d: 2, A: 1, B: 1, C: 1, D: 1, e: 1 }
```

3. Una vez realizado los ejercicios 1 y 2, es hora de cambiar de repo. Para ello lo primero que debes de hacer es crear otro repo en GitHub. Una vez creado el repo haremos lo siguiente, puedes verificar que al ejecutar el comando “git remote” desde la consola de VS Code estando en la raíz del proyecto, aparecerá el mensaje “origin” esto porque ya se tiene una instancia remota.



Ahora haremos lo siguiente, ejecutaremos el comando:

```
C:\Users\Hp\Desktop\Seminario\LAB03-01>git remote add Neworigin  
https://github.com/Victoria-Jimenez-Martinez/Aux_LAB3-Victoria-J  
imenez-Martinez.git
```

```
C:\Users\Hp\Desktop\Seminario\LAB03-01>
```

```
C:\Users\Hp\Desktop\Seminario\LAB03-01>git remote add Neworigin https://github.co  
m/Victoria-Jimenez-Martinez/Aux_LAB3-Victoria-Jimenez-Martinez.git
```

```
C:\Users\Hp\Desktop\Seminario\LAB03-01>git remote  
Neworigin  
origin
```

```
C:\Users\Hp\Desktop\Seminario\LAB03-01>
```

```
C:\Users\Hp\Desktop\Seminario\LAB03-01>git remote remove origin
```

```
C:\Users\Hp\Desktop\Seminario\LAB03-01>git remote  
Neworigin  
origin
```

```
C:\Users\Hp\Desktop\Seminario\LAB03-01>
```

```
C:\Users\Hp\Desktop\Seminario\LAB03-01>git branch -M master
```

```
C:\Users\Hp\Desktop\Seminario\LAB03-01>git Neworigin master  
git: 'Neworigin' is not a git command. See 'git --help'.
```

```
C:\Users\Hp\Desktop\Seminario\LAB03-01>git push -u Neworigin master
```

```
Enumerating objects: 21, done.
```

```
Counting objects: 100% (21/21), done.
```

```
Delta compression using up to 8 threads
```

```
Compressing objects: 100% (14/14), done.
```

```
Writing objects: 100% (21/21), 9.96 KiB | 9.96 MiB/s, done.
```

```
Total 21 (delta 4), reused 21 (delta 4), pack-reused 0
```

```
remote: Resolving deltas: 100% (4/4), done.
```

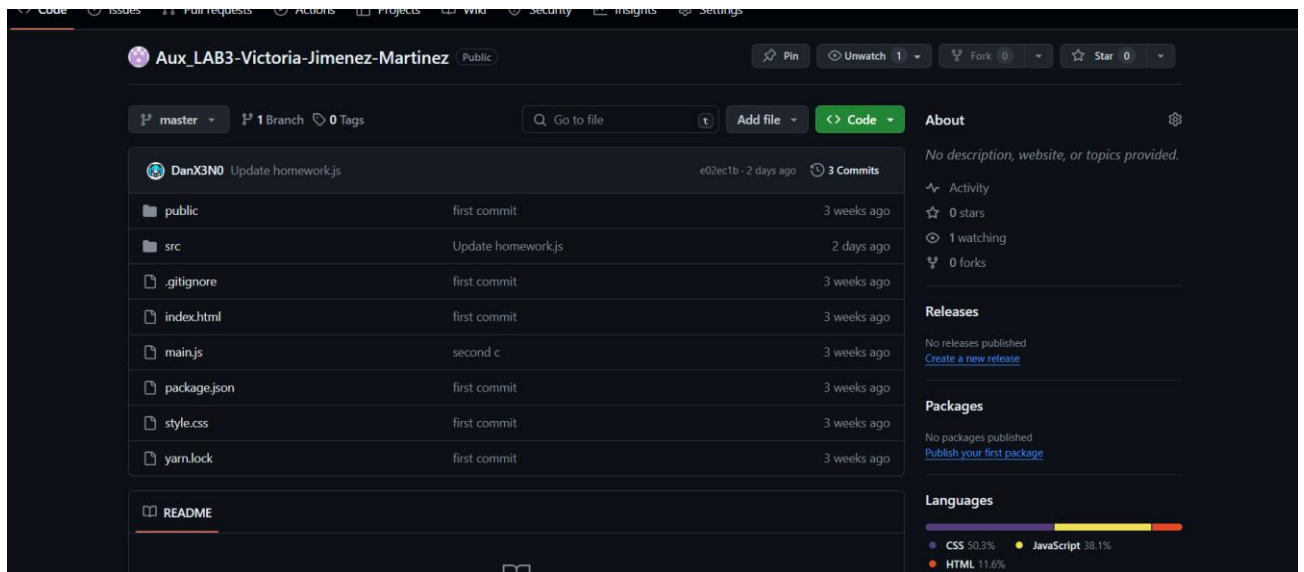
```
To https://github.com/Victoria-Jimenez-Martinez/Aux_LAB3-Victoria-Jimenez-Martine  
z.git
```

```
* [new branch] master -> master
```

```
branch 'master' set up to track Neworigin/master.
```

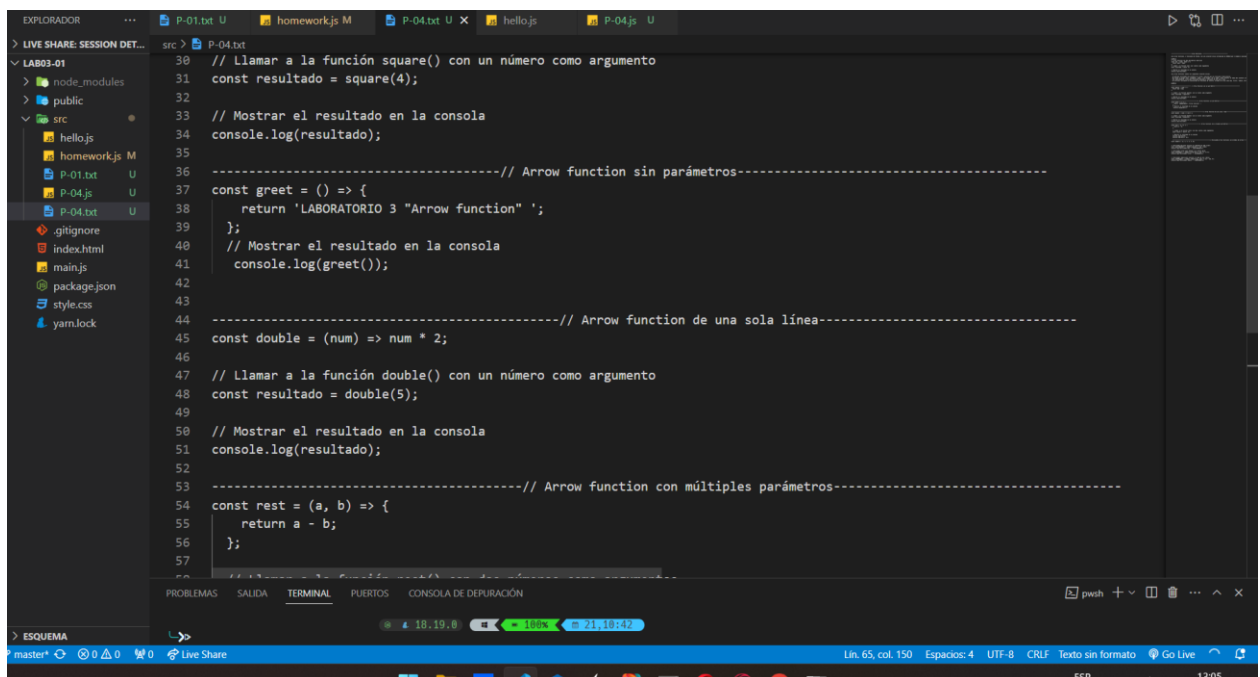
```
C:\Users\Hp\Desktop\Seminario\LAB03-01>
```

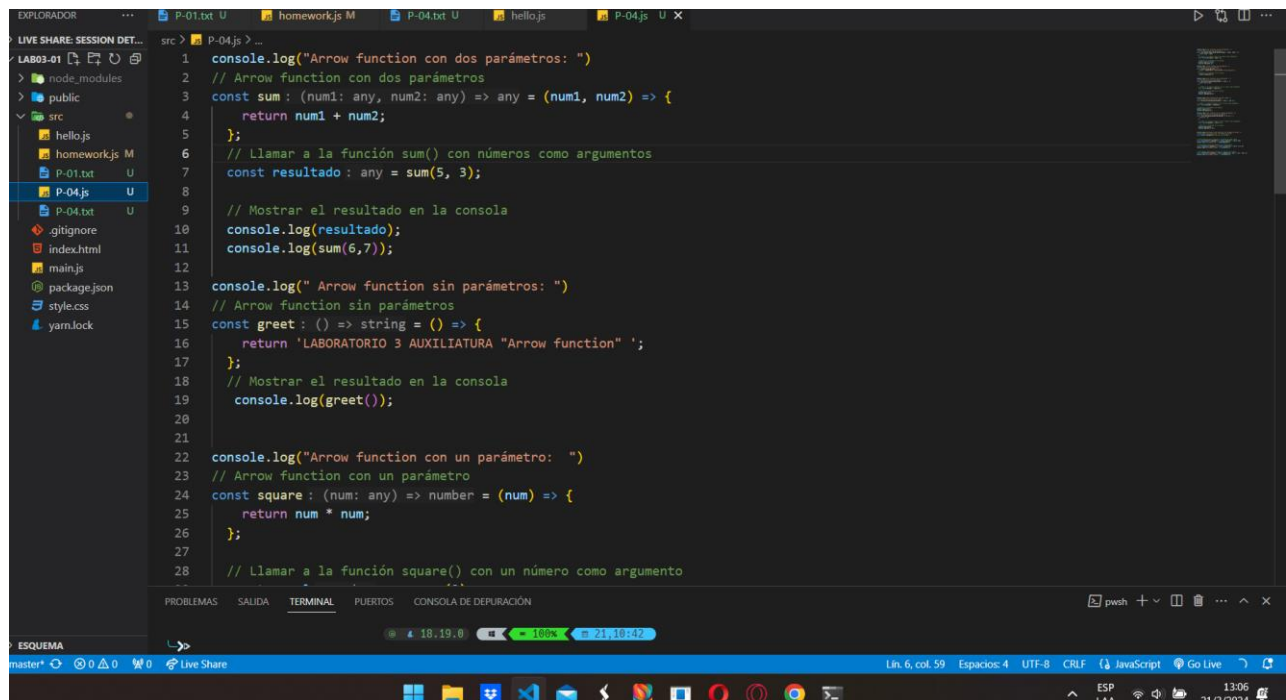




4.- Averigua acerca de las denominadas “arrow functions” y callbacks en javascript. Guardalo en un archivo P-04.txt.

```
C:\Users\Hp\Desktop\Seminario\LAB03-01\src>echo > P-04.txt
C:\Users\Hp\Desktop\Seminario\LAB03-01\src>
```

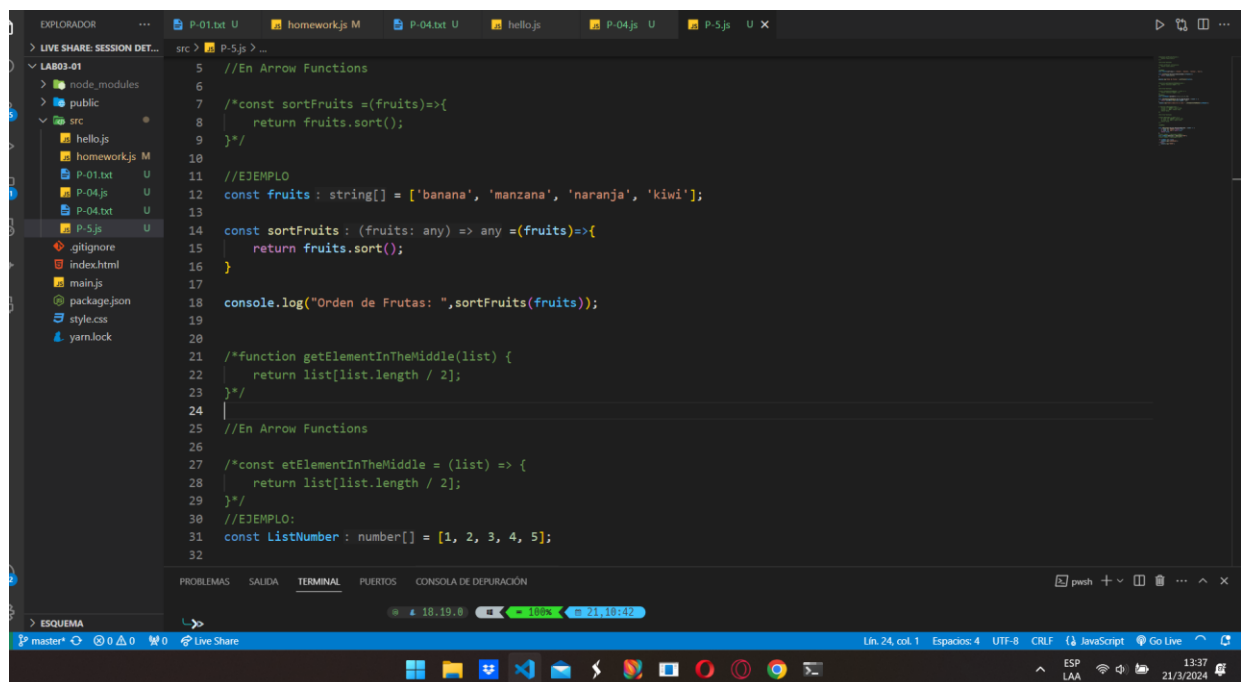




The screenshot shows a VS Code editor window with a file explorer on the left and a code editor in the center. The file explorer shows a project structure with files like P-01.txt, P-04.js, P-04.txt, .gitignore, index.html, main.js, package.json, style.css, and yarn.lock. The code editor displays the content of P-04.js, which contains three arrow functions: a two-parameter function 'sum', a no-parameter function 'greet', and a one-parameter function 'square'. Each function is followed by a call to console.log to demonstrate its use.

```
1 console.log("Arrow function con dos parámetros: ")
2 // Arrow function con dos parámetros
3 const sum : (num1: any, num2: any) => any = (num1, num2) => {
4     return num1 + num2;
5 };
6 // Llamar a la función sum() con números como argumentos
7 const resultado : any = sum(5, 3);
8
9 // Mostrar el resultado en la consola
10 console.log(resultado);
11 console.log(sum(6,7));
12
13 console.log(" Arrow function sin parámetros: ")
14 // Arrow function sin parámetros
15 const greet : () => string = () => {
16     return 'LABORATORIO 3 AUXILIATURA "Arrow function" ';
17 };
18 // Mostrar el resultado en la consola
19 console.log(greet());
20
21
22 console.log("Arrow function con un parámetro: ")
23 // Arrow function con un parámetro
24 const square : (num: any) => number = (num) => {
25     return num * num;
26 };
27
28 // Llamar a la función square() con un número como argumento
```

5.- Ahora haremos uso de las arrow functions, convierte las siguientes funciones en arrow functions. Debera entregarse en un archivo P-05.js.



The screenshot shows a VS Code editor window with a file explorer on the left and a code editor in the center. The file explorer shows a project structure with files like P-01.txt, P-04.js, P-04.txt, .gitignore, index.html, main.js, package.json, style.css, and yarn.lock. The code editor displays the content of P-5.js, which contains three arrow functions: a two-parameter function 'sortFruits', a no-parameter function 'getElementInTheMiddle', and a one-parameter function 'etElementInTheMiddle'. Each function is followed by a call to console.log to demonstrate its use.

```
5 //En Arrow Functions
6
7 /*const sortFruits =(fruits)=>{
8     return fruits.sort();
9 }*/
10
11 //EJEMPLO
12 const fruits : string[] = ['banana', 'manzana', 'naranja', 'kiwi'];
13
14 const sortFruits : (fruits: any) => any =(fruits)=>{
15     return fruits.sort();
16 }
17
18 console.log("Orden de Frutas :",sortFruits(fruits));
19
20
21 /*function getElementInTheMiddle(list) {
22     return list[list.length / 2];
23 }*/
24
25 //En Arrow Functions
26
27 /*const etElementInTheMiddle = (list) => {
28     return list[list.length / 2];
29 }*/
30 //EJEMPLO:
31 const listNumber : number[] = [1, 2, 3, 4, 5];
32
```

```

/*function sortFruits(fruits) {
|   return fruits.sort();
| } */

//En Arrow Functions

/*const sortFruits =(fruits)=>{
|   return fruits.sort();
| }*/

//EJEMPLO
const fruits : string[] = ['banana', 'manzana', 'naranja', 'kiwi'];

const sortFruits : (fruits: any) => any =(fruits)=>{
|   return fruits.sort();
| }

console.log("Orden de Frutas: ",sortFruits(fruits));

```

```

C:\Users\Hp\Desktop\Seminario\LAB03-01\src>node P-5.js
Orden de Frutas: [ 'banana', 'kiwi', 'manzana', 'naranja' ]
Numero medio de la lista: 3
BIENVENIDO

C:\Users\Hp\Desktop\Seminario\LAB03-01\src>node P-5.js
Orden de Frutas: [ 'banana', 'kiwi', 'manzana', 'naranja' ]
Numero medio de la lista: 3
ERROR

C:\Users\Hp\Desktop\Seminario\LAB03-01\src>node P-5.js
Orden de Frutas: [ 'banana', 'kiwi', 'manzana', 'naranja' ]
Numero medio de la lista: 3
BIENVENIDO

C:\Users\Hp\Desktop\Seminario\LAB03-01\src>node P-5.js
Orden de Frutas: [ 'banana', 'kiwi', 'manzana', 'naranja' ]
Numero medio de la lista: 3
BIENVENIDO

C:\Users\Hp\Desktop\Seminario\LAB03-01\src>

```

ENLACE DE GITHUB DEL REPOSITORIO.

[https://github.com/Victoria-Jimenez-Martinez/Aux\\_LAB3-Victoria-Jimenez-Martinez.git](https://github.com/Victoria-Jimenez-Martinez/Aux_LAB3-Victoria-Jimenez-Martinez.git)