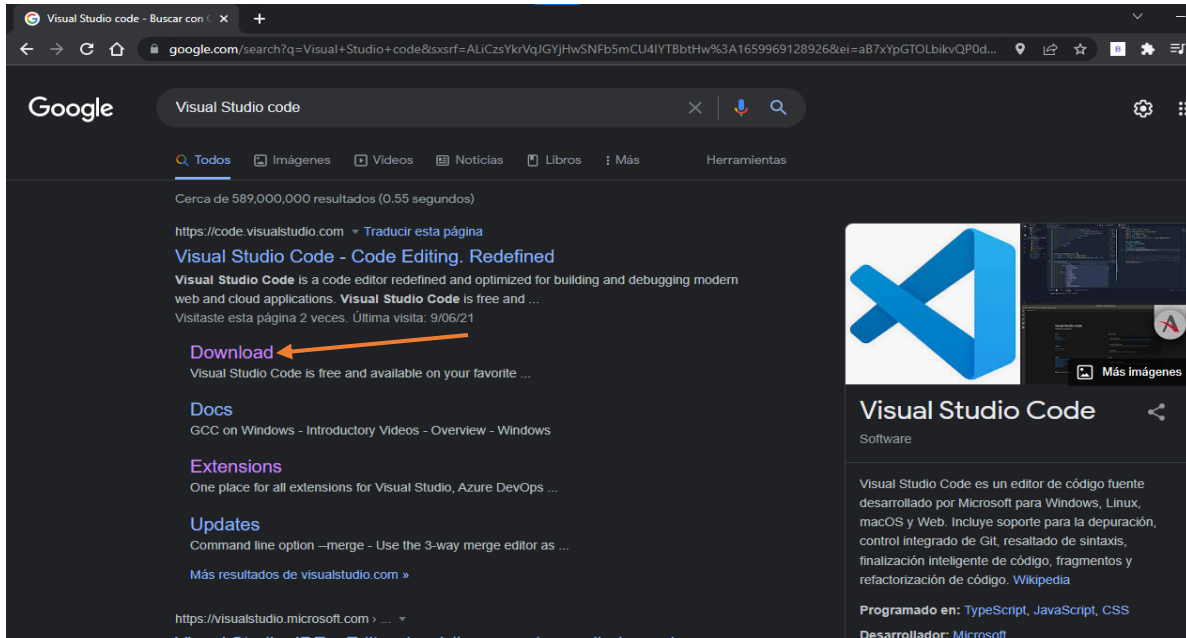
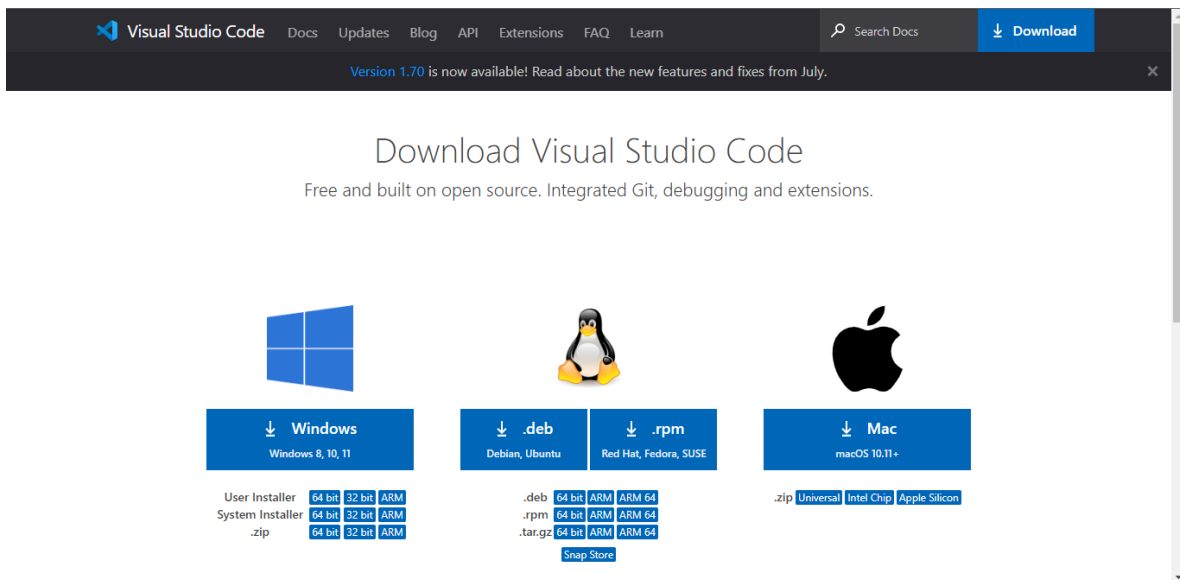


Proceso instalación Visual Studio Code

Paso 1: Escribimos en Google Visual Studio Code y seleccionamos donde dice “Download”.



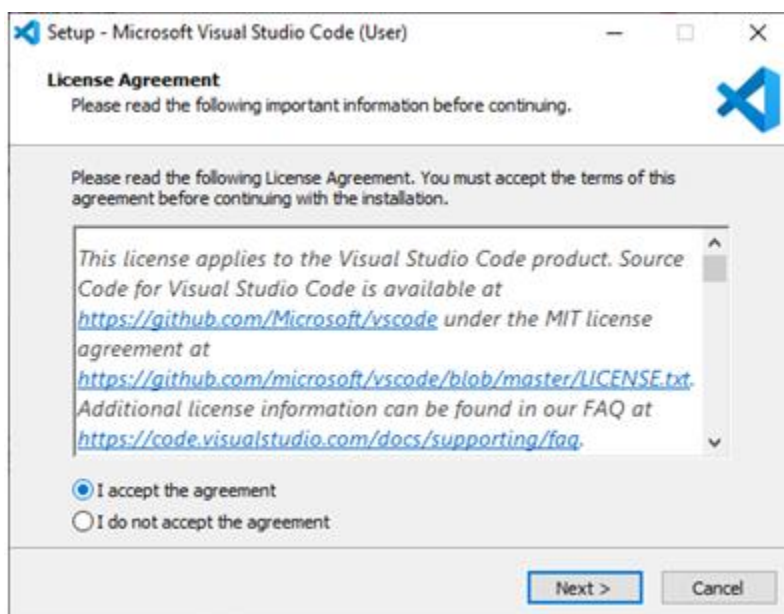
Paso 2: Seleccionamos el sistema operativo que tenemos y lo descargamos.



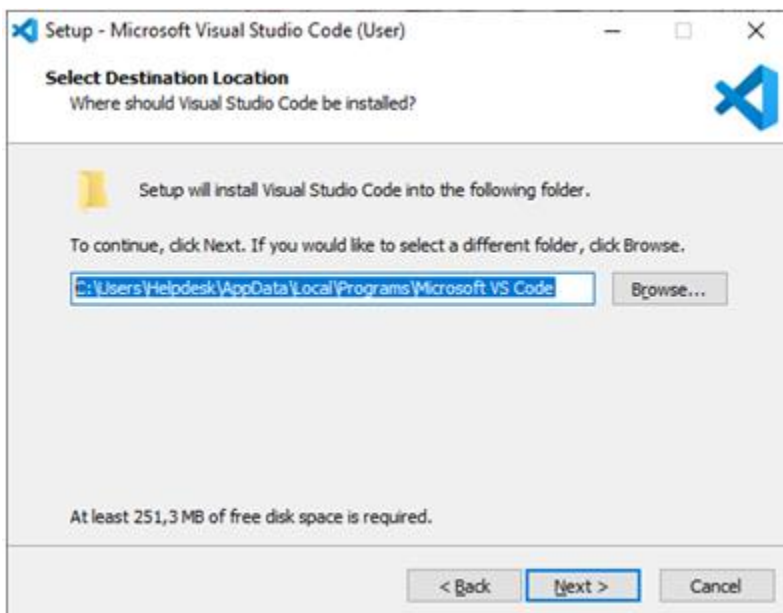
Paso 3: Al darle clic nos descargará un .exe, al cual le daremos clic encima.



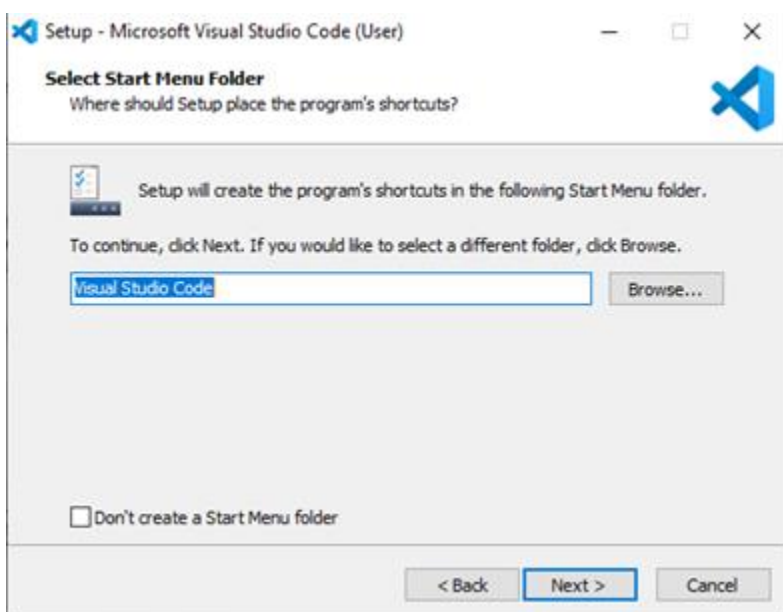
Paso 4: Lee y acepta el acuerdo de licencia. Haz clic en Next para continuar.



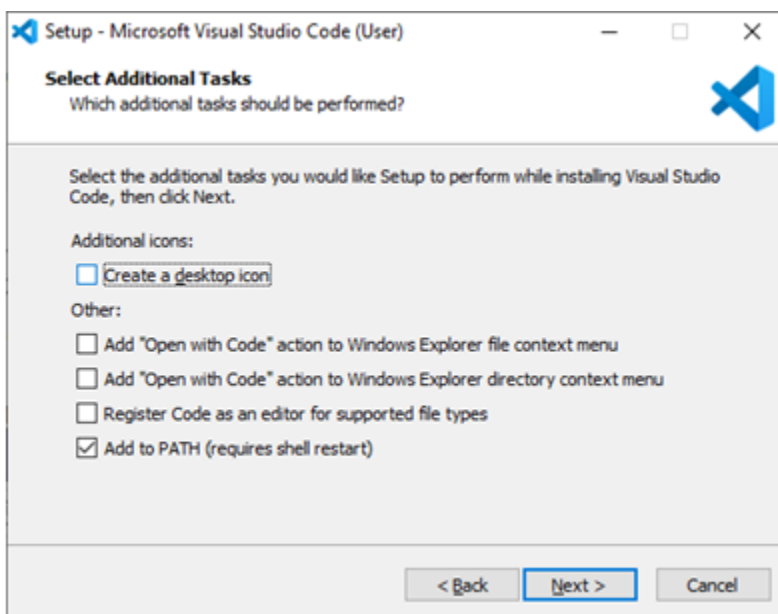
Paso 5: Puedes cambiar la ubicación de la carpeta de instalación o mantener la configuración predeterminada. Haz clic en Next para continuar.



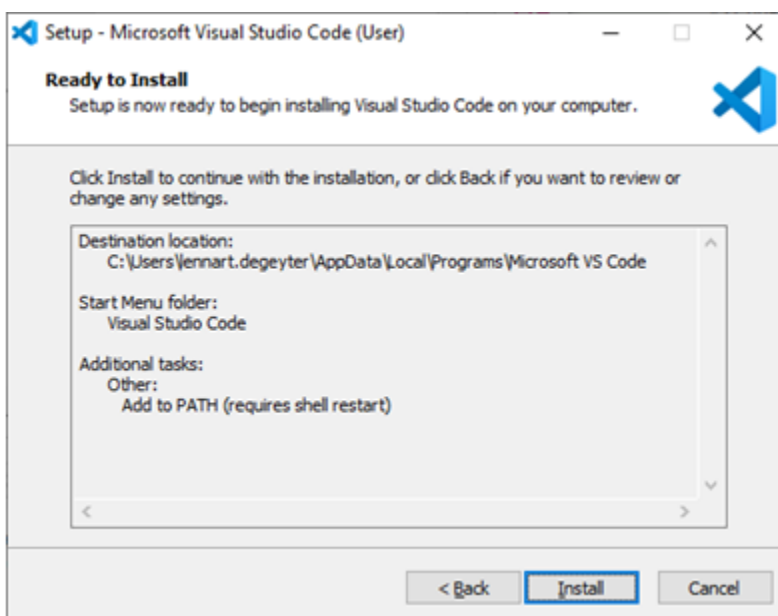
Paso 6: Elige si deseas cambiar el nombre de la carpeta de accesos directos en el menú Inicio o si no deseas instalar accesos directos en absoluto. Haz clic en Next.



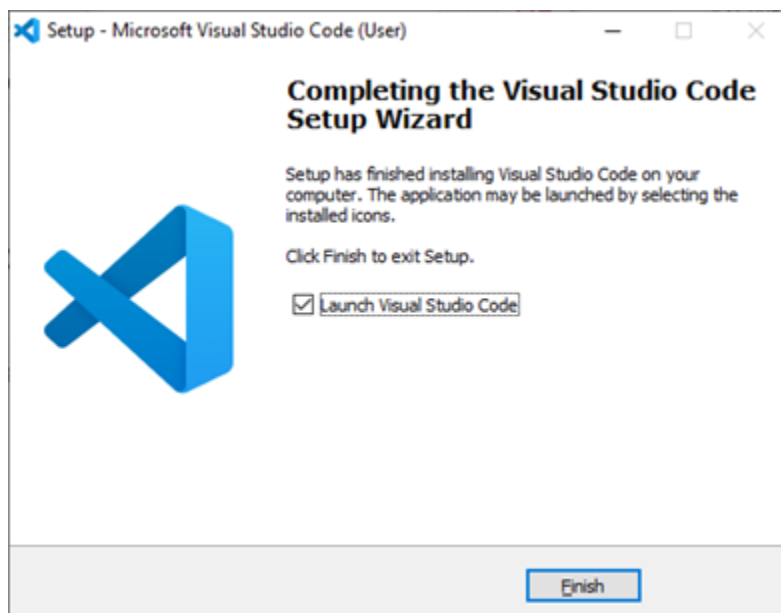
Paso 7: Selecciona las tareas adicionales, por ej. crear un icono en el escritorio o añadir opciones al menú contextual de Windows Explorer. Haz clic en Next.



Paso 8: Haz clic en Install para iniciar la instalación.




Paso 9: El programa está instalado y listo para usar. Haz clic en Finish para finalizar la instalación y lanzar el programa.



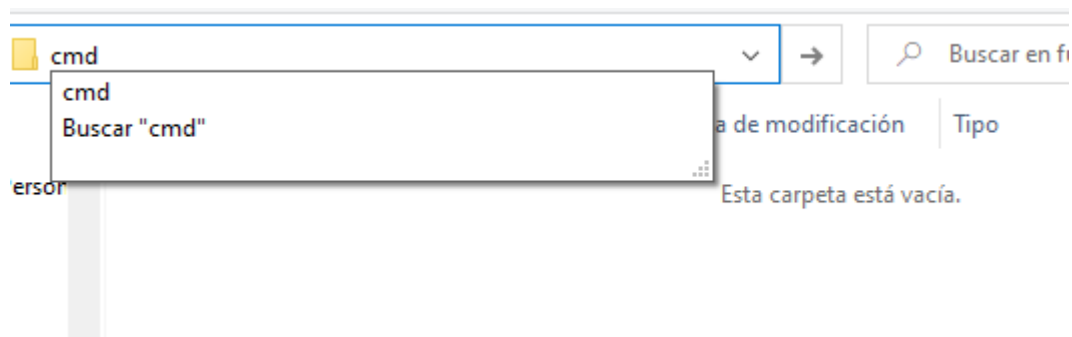
Creación de archivos y carpetas

Paso 1: Creamos una carpeta raíz. En nuestro caso la llamaremos “funcionflecha”.

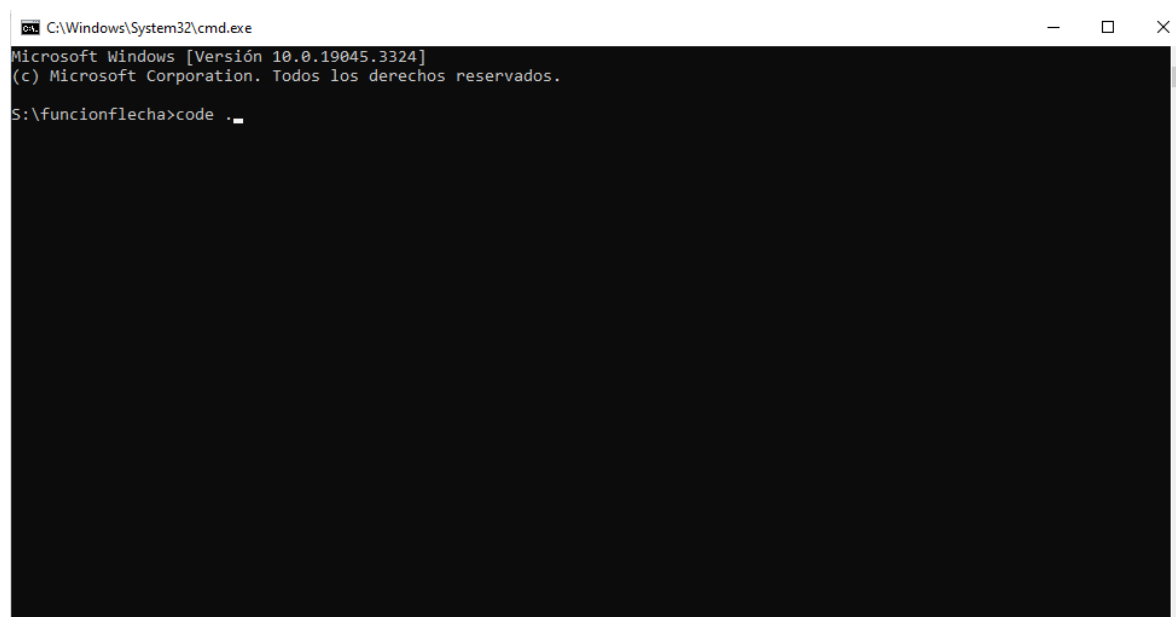
Nombre	Fecha de modificación	Tipo	Tamaño
 funcionflecha	10/09/2023 1:25 a. m.	Carpeta de archivos	

Función Flecha

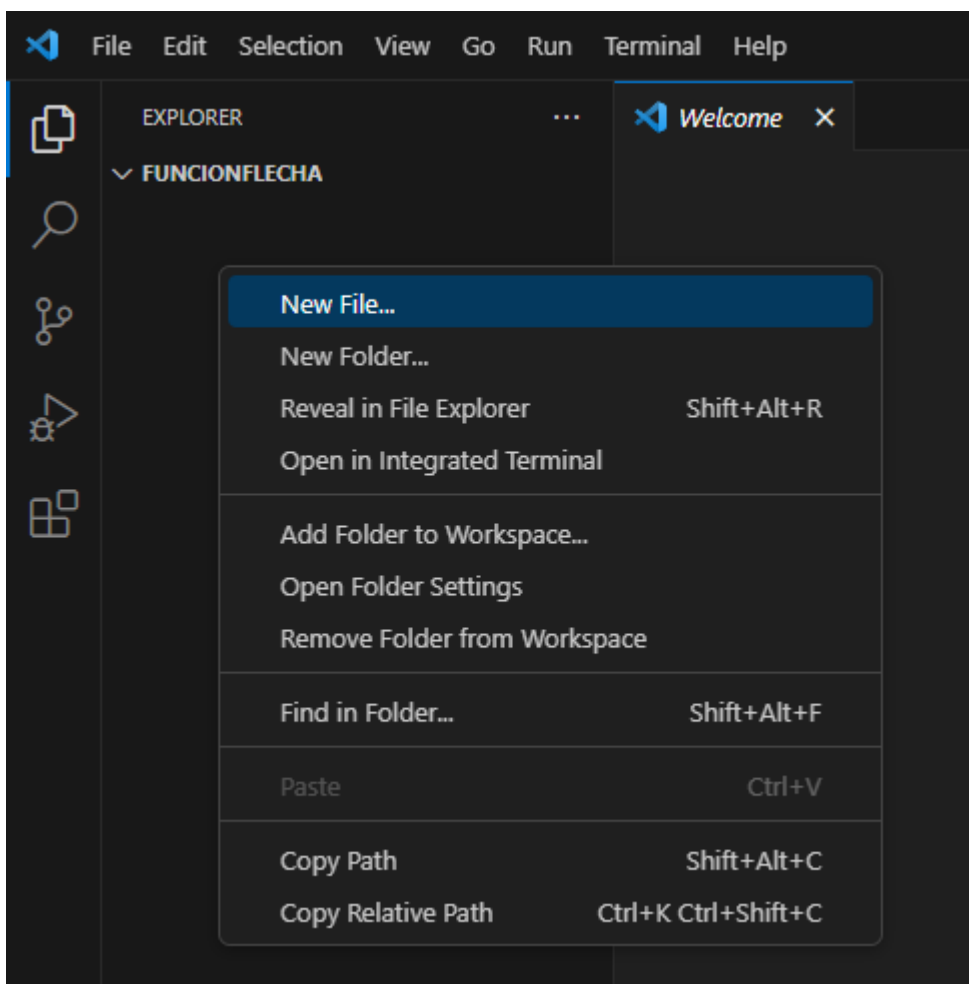
Paso 1: Para abrir el Visual Studio Code, haremos lo siguiente: Dentro de la carpeta raíz, en la barra superior, escribiremos cmd y le damos enter.



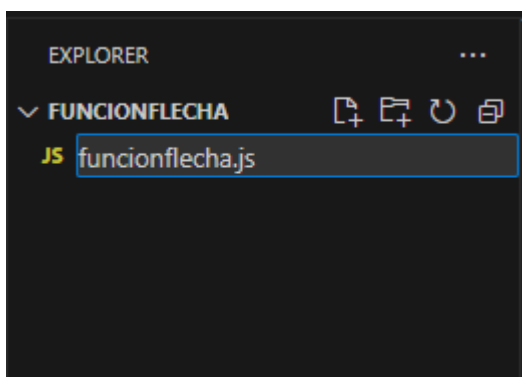
Paso 2: Eso nos abrirá una terminal, solo tendremos que escribir “code .”.



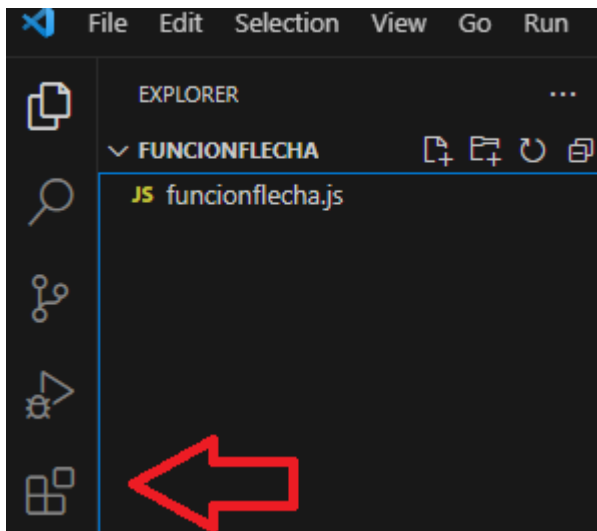
Paso 3: Esto nos abrirá el visual studio code. Ahora, ya que estamos dentro, crearemos un archivo llamado “funciónflecha.js” dándole click derecho a la barra de la izquierda y luego click izquierdo en “new file” o “nuevo archivo”



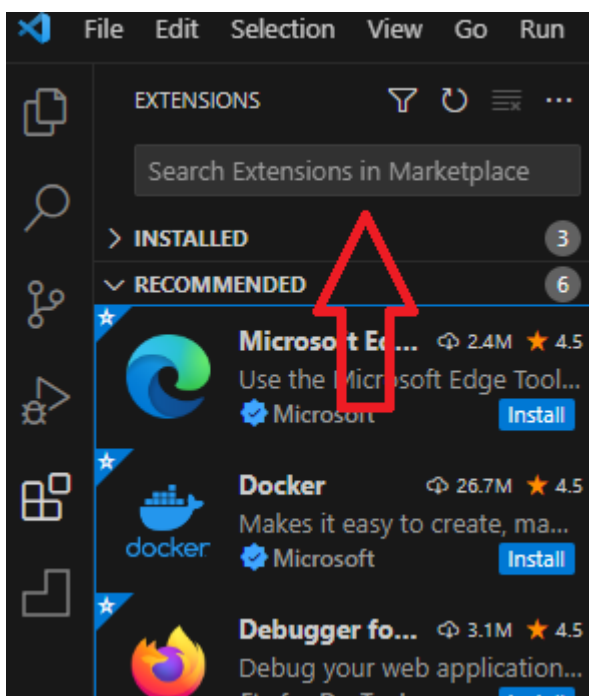
Paso 4: Llamaremos el archivo “funcionflecha.js”



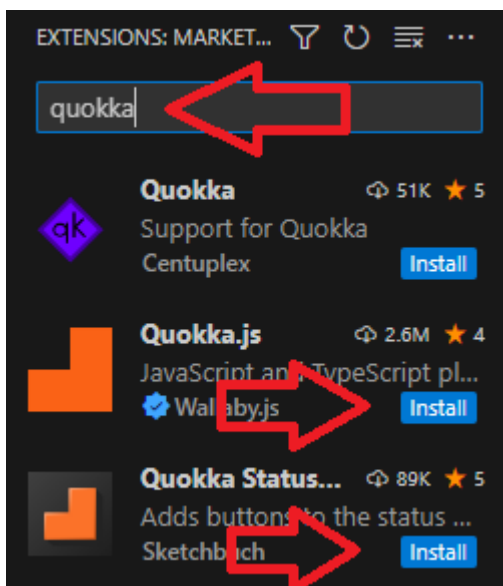
Paso 5: Ahora vamos a instalar algunas extensiones necesarias para mirar nuestra función flecha en la consola, dándole click izquierdo en el símbolo señalado por la flecha



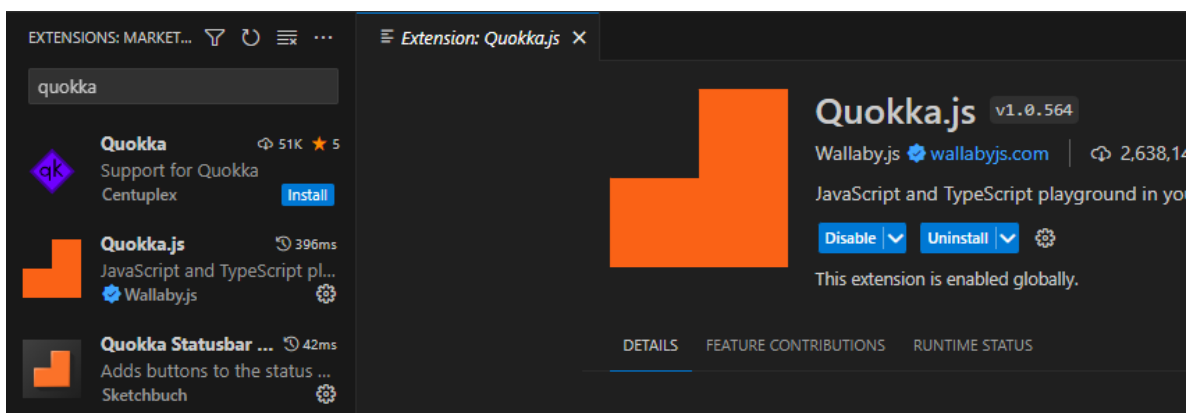
Paso 6: Ahora nos saldrá esta pestaña y en la barra de búsqueda colocamos “quokka”



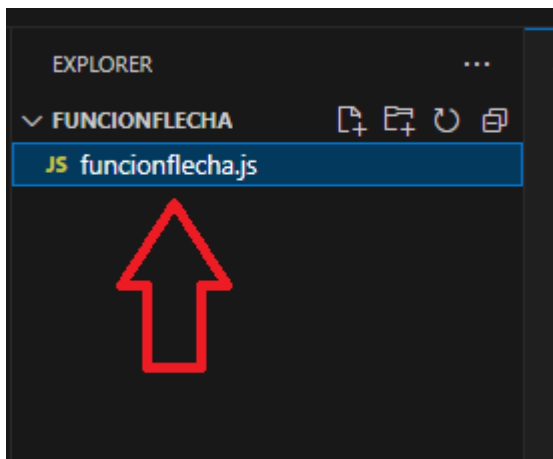
Paso 7: Ahora instalamos los 2 señalados por la flecha



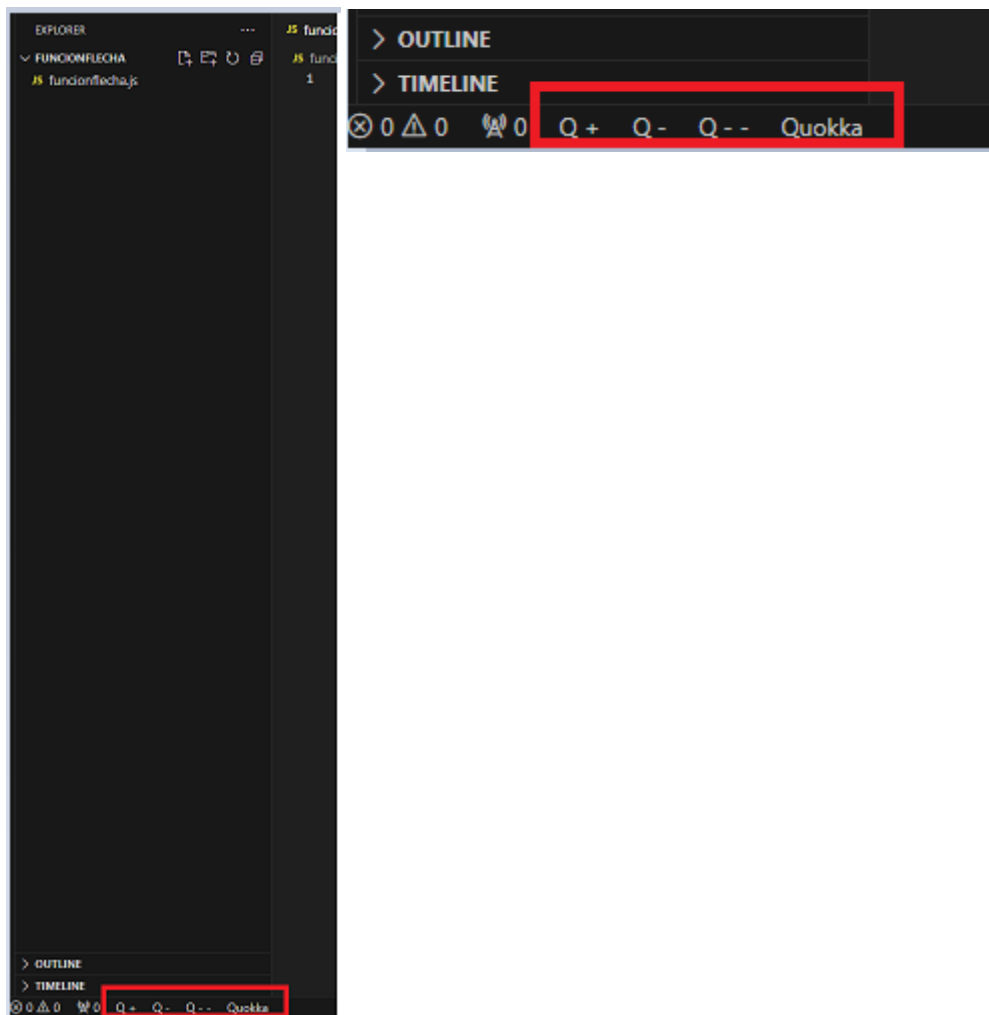
Paso 8: Ahora podemos mirar que no sale la opción para instalar, vamos a reiniciar nuestro visual studio code



Paso 9: Ahora vamos a seleccionar de nuevo nuestro archivo.



Paso 10: En caso de que se haya instalado correctamente nos saldrán estas opciones en la parte inferior



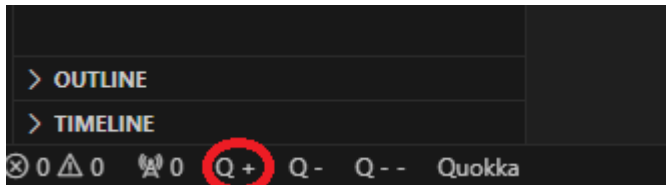
Paso 11: Ahora comenzaremos creando una función normal llamada “nombre” (línea 2) y una función flecha llamada “funcionFlecha” (línea 7) como podemos observar ambos pueden recibir parámetros, pero observamos una diferencia en la sintaxis observando que la función flecha es más utilizada para hacer la sintaxis un poco más corta y predecible omitiendo código como “function”.

```
JS funcionflecha.js > [?] funcionFlecha
1 //Función normal
2 let nombre = function(parametros){
3     //Código a ejecutar
4 }
5
6 //Función flecha
7 let funcionFlecha = (parametros) =>{
8     //Código a ejecutar
9 }
```

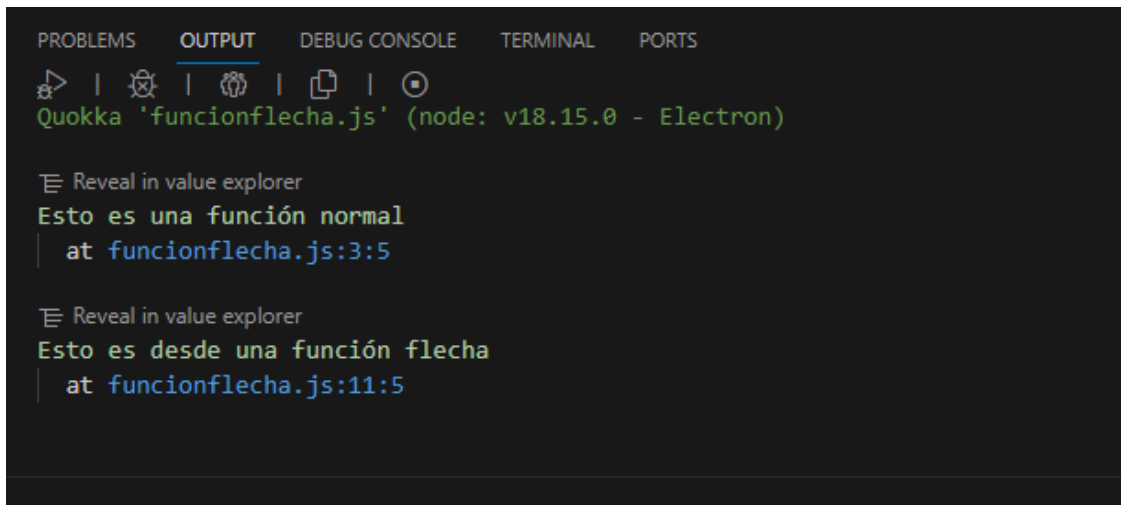
Paso 12: Ahora colocamos código para que se imprima un mensaje en la consola con ambas funciones (línea 3 y línea 11) y además llamaremos a las dos funciones para observarlas en la consola (línea 7 y línea 15)

```
JS funcionflecha.js X
JS funcionflecha.js > ...
1 //Función normal
2 let nombre = function(parametros){
3     console.log('Esto es una función normal')
4 }
5
6 //Llamar función normal
7 nombre ();
8
9 //Función flecha
10 let funcionFlecha = (parametros) =>{
11     console.log('Esto es desde una función flecha')
12 }
13
14 //Llamar función flecha
15 funcionFlecha();
```

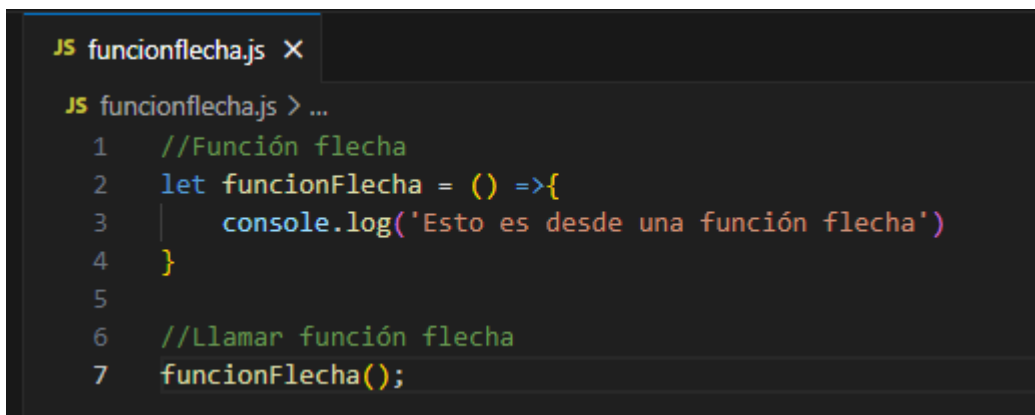
Paso 13: Ahora gracias a las extensiones antes instaladas escogemos la siguiente opción para que se abra la consola.



Paso 14: Así se ve la consola y podemos observar que ambas se imprimen individualmente.



Paso 15: Como nos vamos a centrar en la función flecha vamos a borrar todo lo de la función normal, solo lo hemos puesto para conocer ambas y ver un poco sus diferencias.



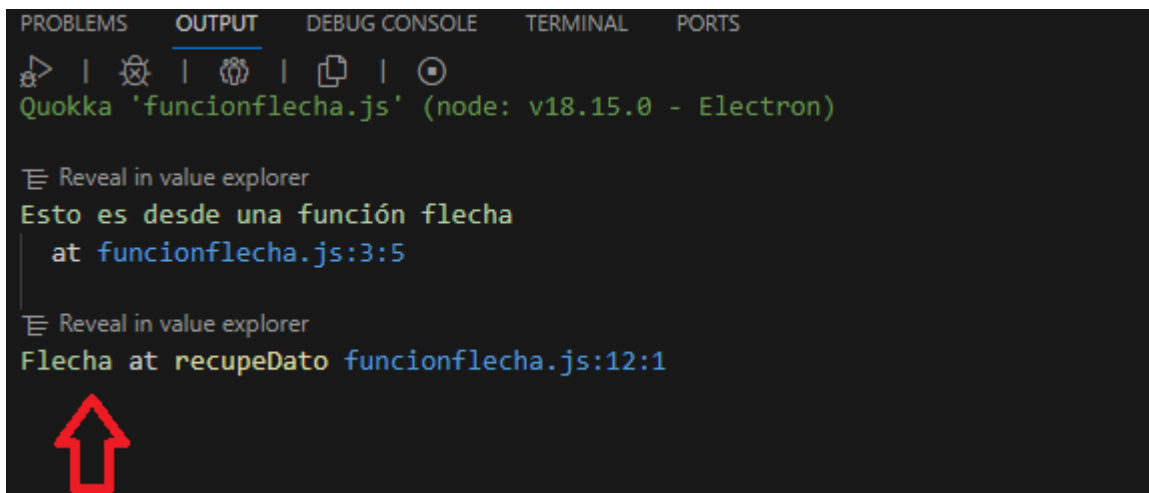
Paso 16: Ahora vamos a hacer que nuestra función flecha retorne un string que diga “flecha” (línea 4)

```
JS funcionflecha.js > ...
1 //Función flecha
2 let funcionFlecha = () =>{
3     console.log('Esto es desde una función flecha')
4     return 'Flecha';
5 }
6
7 //Llamar función flecha
8 funcionFlecha();
```

Paso 17: Ahora vamos a comentar donde llamamos anteriormente la función (línea 8) y vamos a recuperar el dato que hemos decidido retornar anteriormente capturándolo en un dato (línea 11) y lo imprimiremos en la consola (línea 12)

```
JS funcionflecha.js X
JS funcionflecha.js > ...
1 //Función flecha
2 let funcionFlecha = () =>{
3     console.log ('Esto es desde una función flecha')
4     return 'Flecha';
5 }
6
7 //Llamar función flecha
8 //funcionFlecha();
9
10 //Recuperar dato de la función
11 let recupeDato = funcionFlecha();
12 console.log (recupeDato);
```

Paso 18: En nuestra consola podremos observar como muestra el dato que le pusimos a retornar

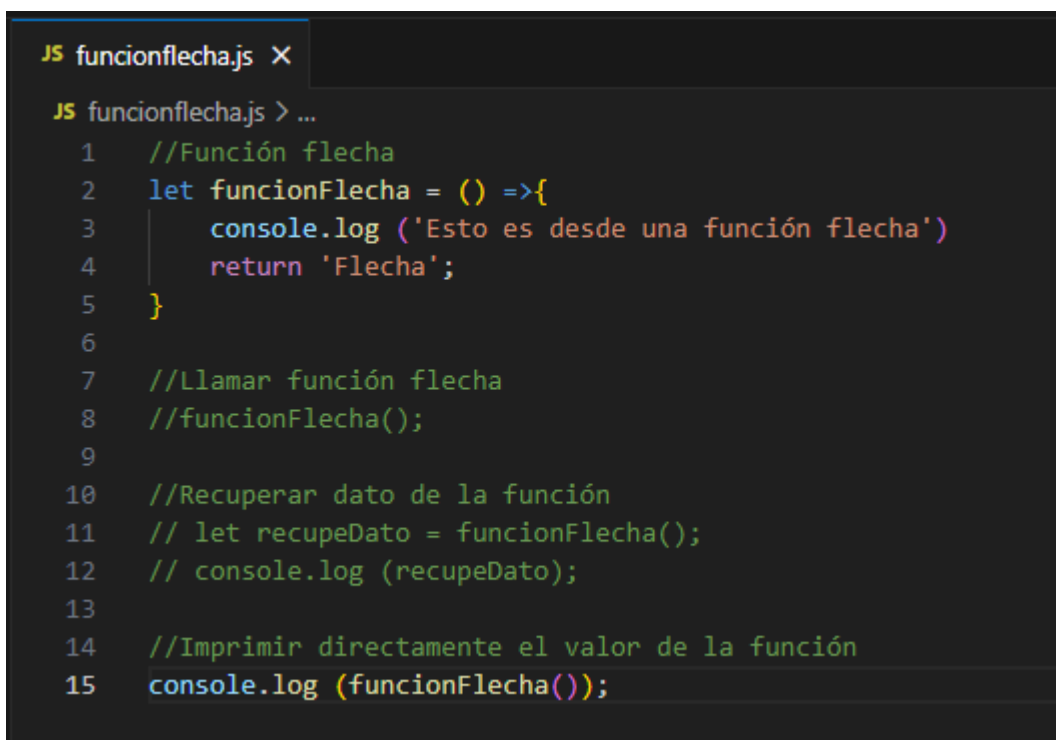


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
Quokka 'funcionflecha.js' (node: v18.15.0 - Electron)

Reveal in value explorer
Esto es desde una función flecha
  at funcionflecha.js:3:5

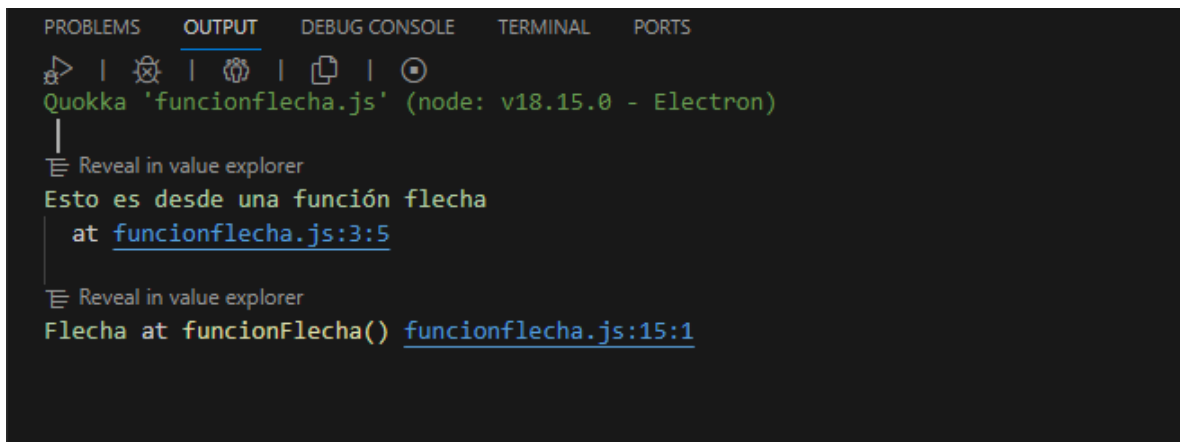
Reveal in value explorer
Flecha at recupeDato funcionflecha.js:12:1
```

Paso 19: Hay otra manera de hacerlo, lo primero que haremos será comentar lo anterior (línea 11 y línea 12) y vamos a imprimir en la consola directamente el valor de funcionFlecha (línea 15)



```
JS funcionflecha.js X
JS funcionflecha.js > ...
1  //Función flecha
2  let funcionFlecha = () =>{
3      console.log ('Esto es desde una función flecha')
4      return 'Flecha';
5  }
6
7  //Llamar función flecha
8  //funcionFlecha();
9
10 //Recuperar dato de la función
11 // let recupeDato = funcionFlecha();
12 // console.log (recupeDato);
13
14 //Imprimir directamente el valor de la función
15 console.log (funcionFlecha());
```

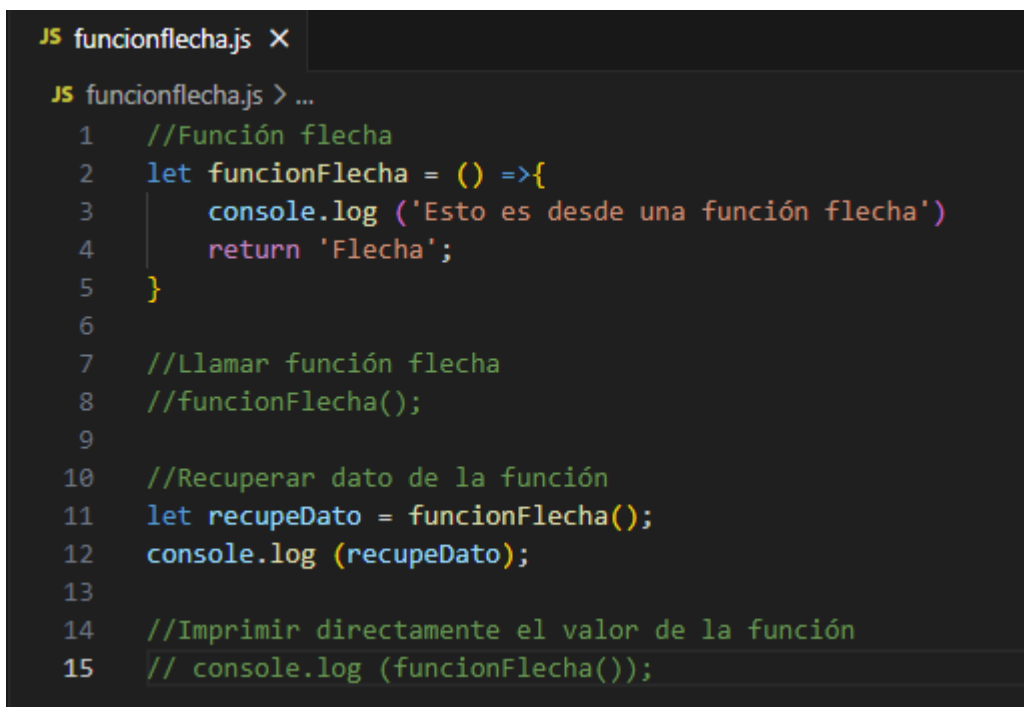
Paso 20: Y observamos que se imprime correctamente en consola



The screenshot shows the Quokka.js output window with the following content:

```
Quokka 'funcionflecha.js' (node: v18.15.0 - Electron)
|
| Te Reveal in value explorer
Esto es desde una función flecha
| at funcionflecha.js:3:5
|
| Te Reveal in value explorer
Flecha at funcionFlecha() funcionflecha.js:15:1
```

Paso 21: Sin embargo, esta última opción no es algo correcto ya que no se podrá llamar debajo de nuevo a menos que coloquemos todo de nuevo, ya que no esta almacenada la respuesta en una variable. Así que vamos a comentar de nuevo lo que acabamos de hacer (línea 15) y vamos a des comentar lo anterior (línea 11 y 12)



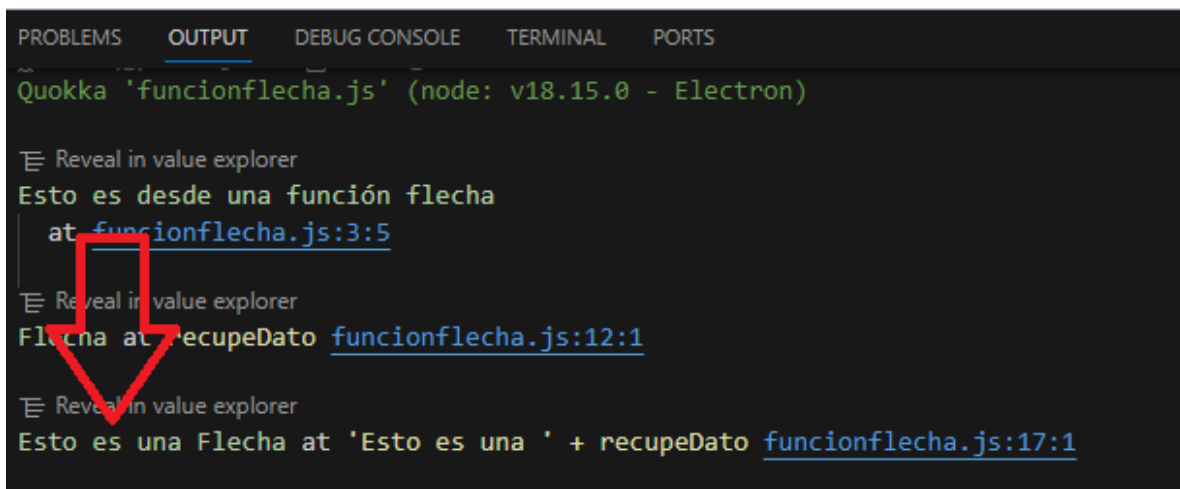
The screenshot shows the source code of 'funcionflecha.js' in the VS Code editor:

```
JS funcionflecha.js X
JS funcionflecha.js > ...
1 //Función flecha
2 let funcionFlecha = () =>{
3     console.log ('Esto es desde una función flecha')
4     return 'Flecha';
5 }
6
7 //Llamar función flecha
8 //funcionFlecha();
9
10 //Recuperar dato de la función
11 let recupeDato = funcionFlecha();
12 console.log (recupeDato);
13
14 //Imprimir directamente el valor de la función
15 // console.log (funcionFlecha());
```


Paso 22: Ahora que tenemos de nuevo el dato almacenado en una variable vamos a podemos hacer cosas como concatenarla de la siguiente manera (línea 17)

```
1 //Función flecha
2 let funcionFlecha = () =>{
3     console.log ('Esto es desde una función flecha')
4     return 'Flecha';
5 }
6
7 //Llamar función flecha
8 //funcionFlecha();
9
10 //Recuperar dato de la función
11 let recupeDato = funcionFlecha();
12 console.log (recupeDato);
13
14 //Imprimir directamente el valor de la función
15 // console.log (funcionFlecha());
16
17 console.log('Esto es una ' + recupeDato);
```

Paso 23: Ahora podemos observar como en la consola imprime todo ya concatenado.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
Quokka 'funcionflecha.js' (node: v18.15.0 - Electron)
⌵ Reveal in value explorer
Esto es desde una función flecha
  at funcionflecha.js:3:5
⌵ Reveal in value explorer
Flecha at recupeDato funcionflecha.js:12:1
⌵ Reveal in value explorer
Esto es una Flecha at 'Esto es una ' + recupeDato funcionflecha.js:17:1
```

Paso 24: Esta prueba no la realices es simplemente para que veas que si tratas de llamar una función flecha (línea 2) antes de ser inicializada (línea 6) enviará un error así que debes tener mucho cuidado de primero inicializar las funciones y luego llamarlas.

```
JS funcionflecha.js X
JS funcionflecha.js > [?] funcionFlecha
1 //Recuperar dato de la función
2 let recupeDato = funcionFlecha(); Cannot access 'funcionFlecha' before initialization
3 console.log (recupeDato);
4
5 //Función flecha
6 let funcionFlecha = () =>{
7     console.log ('Esto es desde una función flecha')
8     return 'Flecha';
9 }
10
11 //Llamar función flecha
12 //funcionFlecha();
13
14
15
16 //Imprimir directamente el valor de la función
17 // console.log (funcionFlecha());
18
19 console.log('Esto es una ' + recupeDato);
```

Paso 25: Ahora te explicaré la el por qué la función flecha es una variable y no una constante.

```
JS funcionflecha.js > [?] funcionFlecha
1 //Función flecha
2 let funcionFlecha = () =>{
3     console.log ('Esto es desde una función flecha')
4     return 'Flecha';
5 }
6
```

Paso 26: Como podemos ver abajo podemos cambiar lo que retorna la funcionFlecha (línea 20) y llamarla de nuevo (línea 24)

```
JS funcionflecha.js > ...
1 //Función flecha
2 let funcionFlecha = () =>{
3     console.log ('Esto es desde una función flecha')
4     return 'Flecha';
5 }
6
7 //Llamar función flecha
8 //funcionFlecha();
9
10 //Recuperar dato de la función
11 let recupeDato = funcionFlecha();
12 console.log (recupeDato);
13
14 //Imprimir directamente el valor de la función
15 // console.log (funcionFlecha());
16
17 console.log('Esto es una ' + recupeDato);
18
19 //Cambio de la función flecha gracias a ser let
20 funcionFlecha = () =>{
21     console.log('Esto es una prueba');
22 }
23
24 funcionFlecha();
```

Paso 27: Y en la consola podemos ver cómo, aunque llame de nuevo a la función flecha lo que retorna ya está cambiado.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Esto es desde una función flecha
  at funcionflecha.js:3:5
  Reveal in value explorer
Flecha at recupeDato funcionflecha.js:12:1
  Reveal in value explorer
Esto es una Flecha at 'Esto es una ' + recupeDato funcionflecha.js:17:1
  Reveal in value explorer
Esto es una prueba at funcionflecha.js:21:5
```

Paso 28: Sin embargo, si colocamos la “funcionFlecha” como una constante y tratamos de cambiar la función abajo nos enviará un error en la consola.

```
JS funcionflecha.js X
JS funcionflecha.js > ...
1 //Función flecha
2 const funcionFlecha = () =>{
3   console.log ('Esto es desde una función flecha')
4   return 'Flecha';
5 }
6
7 //Llamar función flecha
8 //funcionFlecha();
9
10 //Recuperar dato de la función
11 let recupeDato = funcionFlecha();
12 console.log (recupeDato);
13
14 //Imprimir directamente el valor de la función
15 // console.log (funcionFlecha());
16
17 console.log('Esto es una ' + recupeDato);
18
19 //Cambio de la función flecha gracias a ser let
20 funcionFlecha = () =>{
21   console.log('Esto es una prueba');
22 }
23
24 funcionFlecha();
```

Paso 29: Como podemos ver en la consola el error que envía así que se podría decir que al declarar una función flecha con let, puedes reasignarla usando let nuevamente después de su declaración. Por otro lado, si declaras una función flecha con const, no puedes reasignarla a otra función o valor después de su declaración. La elección entre let y const depende de si necesitas o no reasignar la función y de si deseas evitar accidentalmente su reasignación en tu código.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Quokka 'funcionflecha.js' (node: v18.15.0 - Electron)

Assignment to constant variable.
at funcionflecha.js:20:1
```

Paso 30: Así que ponemos de nuevo la función flecha como variable y ahora ya conoces la función flecha y algunas de sus funciones!

```
JS funcionflecha.js X
JS funcionflecha.js > ...
1 //Función flecha
2 let funcionFlecha = () =>{
3     console.log ('Esto es desde una función flecha')
4     return 'Flecha';
5 }
6
7 //Llamar función flecha
8 //funcionFlecha();
9
10 //Recuperar dato de la función
11 let recupeDato = funcionFlecha();
12 console.log (recupeDato);
13
14 //Imprimir directamente el valor de la función
15 // console.log (funcionFlecha());
16
17 console.log('Esto es una ' + recupeDato);
18
19 //Cambio de la función flecha gracias a ser let
20 funcionFlecha = () =>{
21     console.log('Esto es una prueba');
22 }
23
24 funcionFlecha();
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
Esto es desde una función flecha
  at funcionflecha.js:3:5
  Reveal in value explorer
Flecha at recupeDato funcionflecha.js:12:1
  Reveal in value explorer
Esto es una Flecha at 'Esto es una ' + recupeDato funcionflecha.js:17:1
  Reveal in value explorer
Esto es una prueba at funcionflecha.js:21:5
```