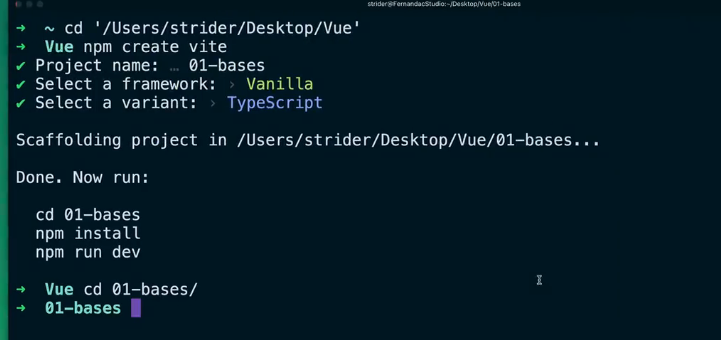
Corre esto



Npm install

Npm run dev

Todo con typescript esta los files terminan en .ts

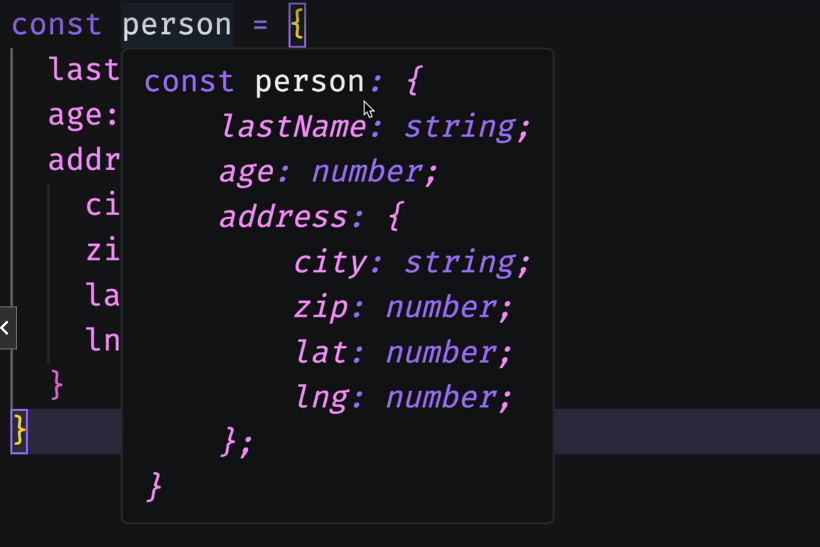
Acá se podría almacenar cualquier tipo de dato tipo string adentro

A black background with white text

Description automatically generated

NO HAY QUE USAR VAR SOLO LET O CONST.

Typescript se da cuenta los tipos solo



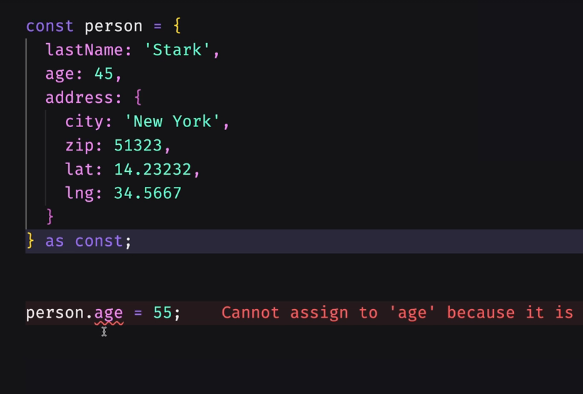
A computer screen shot of a computer code

Description automatically generated

Esto aplica restricciones

Así se puede modificar los valores dentro de person por ejemplo, podes hacer person.age = 35, eso es valido mas allá de que es una constante, lo que no se puede hacer es person = {}

Si queremos que no se modifique nada de dentro hacer esto:

A computer screen shot of a code

Description automatically generated

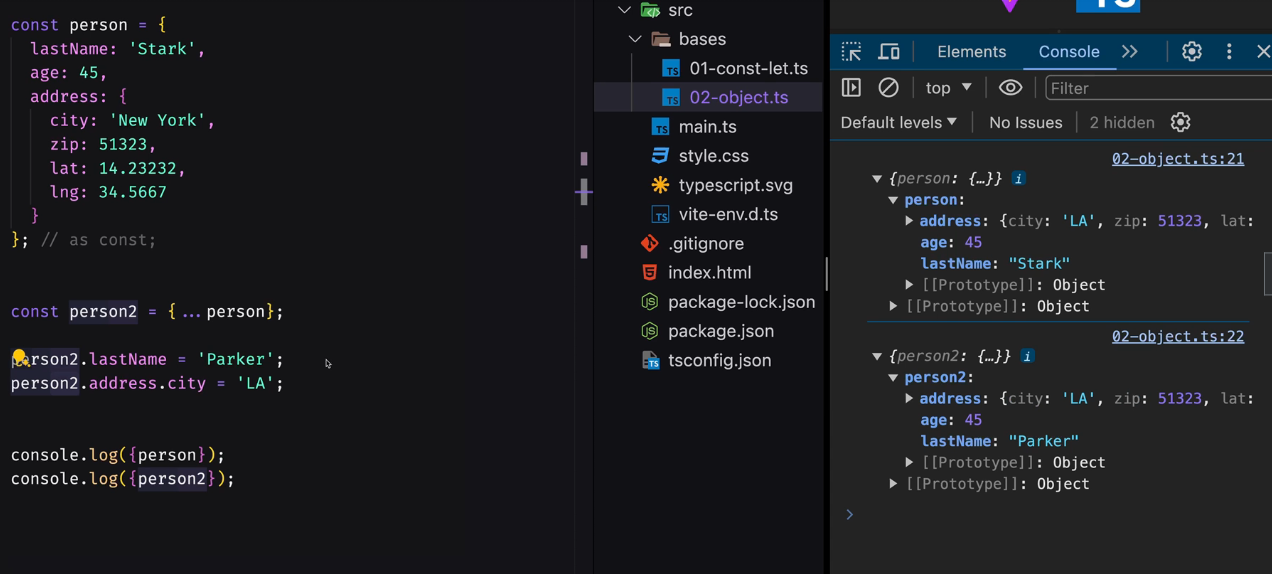
Acá como se esta pasando la referencia, se van a cambiar los dos a Parker. Todos los objetos en js se pasan por referencia

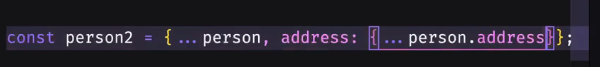
A screenshot of a computer

Description automatically generated

Los otros elementos no, para crear una copia se hace esto:   


Pero esto no es del todo real. Que pasa si hay un objeto adentro del objeto? Porque si modificamos adress city también se va a aplicar a los dos



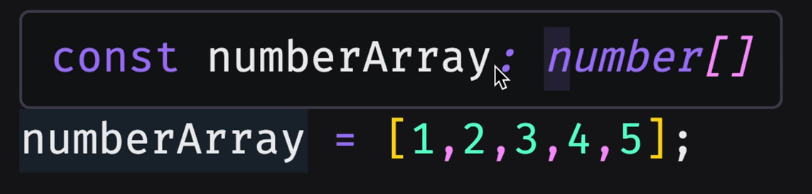
Sería tedioso y habría que hacer esto:   


Una mejor manera de hacerlo es así: usando structuredClone()

A close up of a computer screen

Description automatically generated

Arrays:



Si no quiere que se edite

A black background with colorful text

Description automatically generated

A screenshot of a computer

Description automatically generated

Mismo problema de antes si se iguala pasa la referencia y se aplica a los dos

A screenshot of a computer

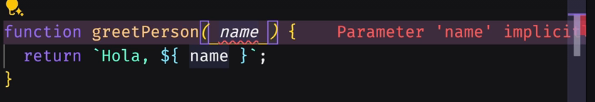
Description automatically generated

Si necesitas que un array acepte números o strings se hace así:   
A black background with purple text

Description automatically generated

Funciones:

Si no se sabe el tipo de dato te lo toma como any, el any es un tipo de dato que hay que tratar de evitar.



Le definimos ahí el tipo



En una función flecha también

Params de tipo string y => void es que no regresa nada

A computer screen shot of a computer

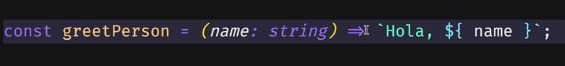
Description automatically generated

Si retornas string te dice que retorna string

A screen shot of a computer

Description automatically generated

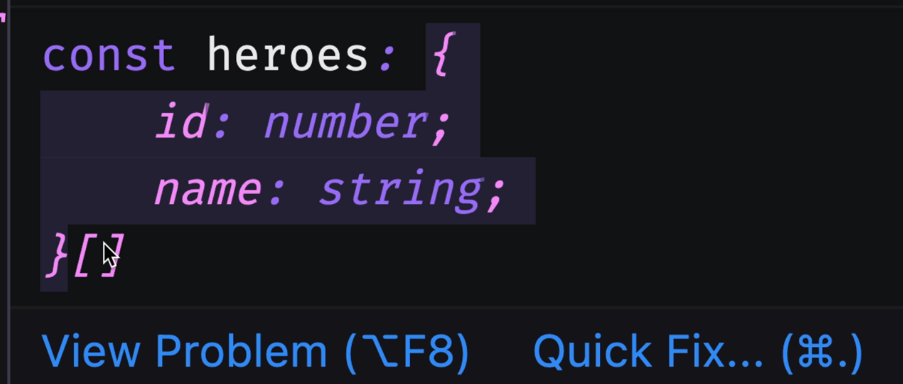
Si la función tiene un solo return se puede eliminar las llaves y el return



Para retornar directamente un objeto se pone entre paréntesis

A black screen with colorful text

Description automatically generated

Si tenemos un array de objetos aparece así:   
  


Que pasa si tenemos valores distintos?   
A screen shot of a computer

Description automatically generated

Aparece ahora como opcional power

A screenshot of a computer program

Description automatically generated

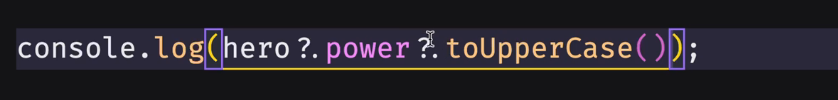
Acá al final el hero puede ser nulo por eso le pone ese signo de interrogación

A computer screen with text and symbols

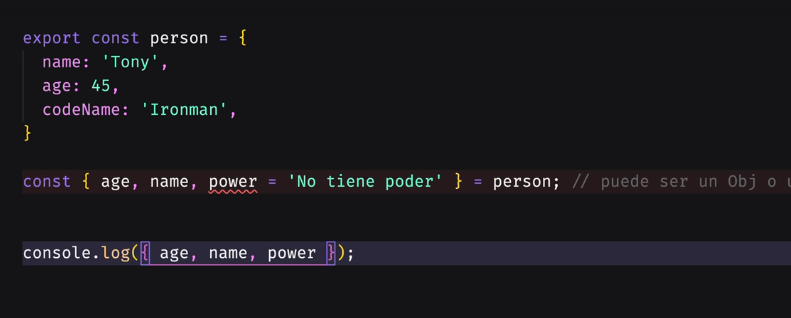
Description automatically generated

Esto va a asegurar de que el código no rompa si ese valor es null

Si se agrega otra función se agregaría otro signo de pregunta



Se puede establecer un valor por defecto para destructuring, pero a typescript no le gusta



Interface son para explicar que estructura tienen los objetos, se puede usar la interfaz como un tipo de datos después.

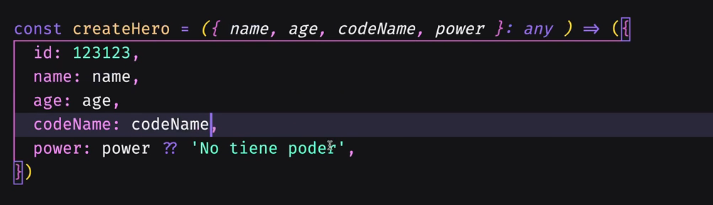
A screen shot of a computer

Description automatically generated

Nueva manera para preguntar en js si algo tiene un valor nulo



Desestructuración en argumentos



Desestructuracion de arrays:

En los arrays se desestructura la posición

A screenshot of a computer

Description automatically generated

Si queres solo el segundo se pone una coma

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

Tipado de datos:

Tenemos un interface y un array

En la interfaz se podría definir algo así:   
A screen shot of a computer

Description automatically generated

Peropodemos definir dc y Marvel en un tipo

A screen shot of a computer

Description automatically generated

Lo podemos usar tonces como tipo de dato

A screenshot of a computer program

Description automatically generated

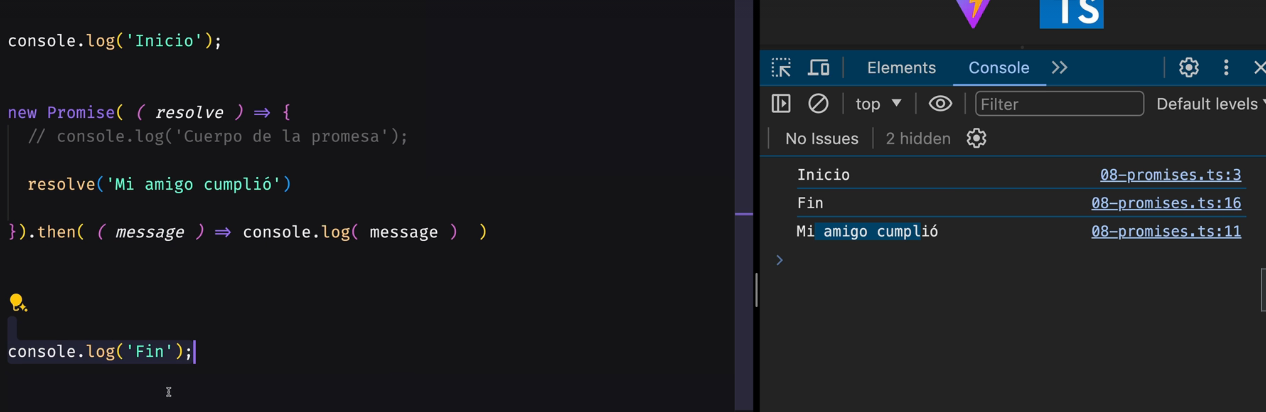
Agregarle el type a las importaciones de interfaces o types



Promesas:

En las promesas el then se va a ejecutar cuando se ejecute el resolve y el catch se ejecutaría cuando se ejecuta el reject y el finally se ejecuta siempre

Prestar atención en cuando es que se ejecuta



Js, las promesas se ejecutan después.

Se podría hacer algo así tonces:   
A screenshot of a computer

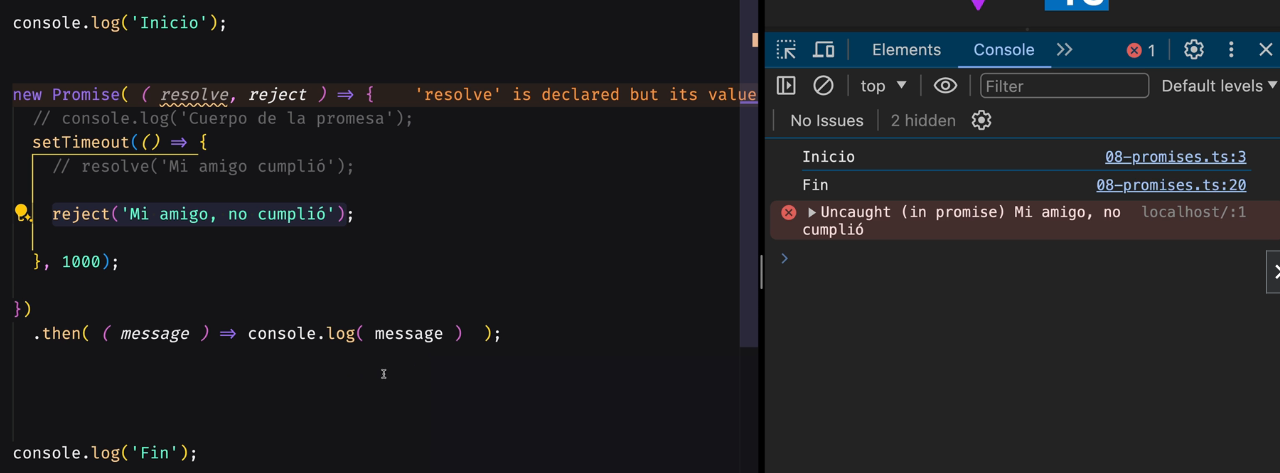
Description automatically generated

Solo se puede llamar una vez el resolve o el reject en la promesa

A screenshot of a computer

Description automatically generated

Este error sale cuando hay una excepción no controlada, osaea se llama al reject pero no hay catch



A screenshot of a computer

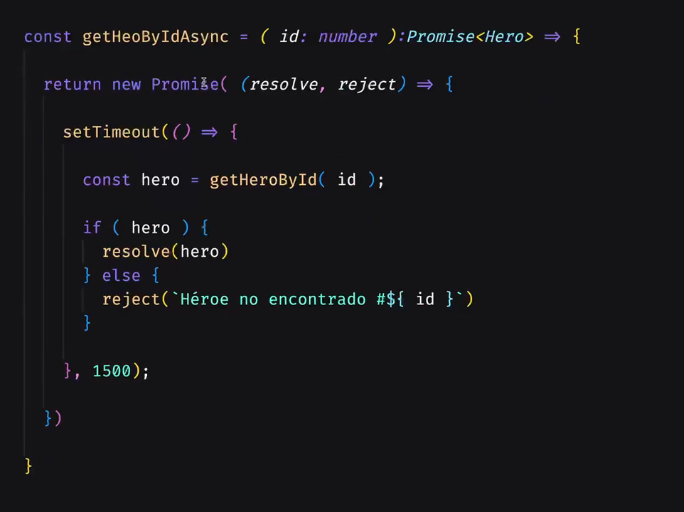
Description automatically generated

A screenshot of a computer

Description automatically generated

Argumentos en las promesas:

Para establecer que retorna una promesa se pone :Promise<tipoDato>

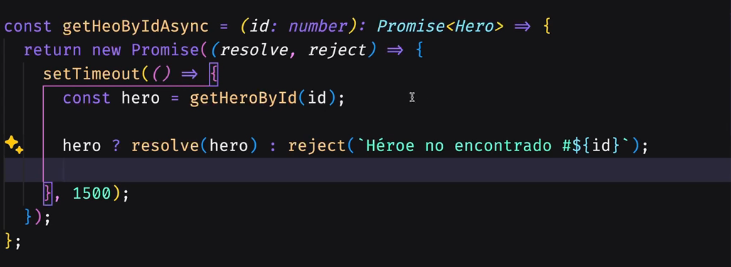


Para llamar a la función promesa sería:

A screen shot of a message

Description automatically generated

Limpiamos la función



API:

Cuando haces una llamada a una api no hay forma de saber que tipo de datos devuelve el body entonces no podemos saber con typescript si esta bien escrito o no, para esto habría que crear una interface.

Podemos agarrar el objeto de respuesta desde postman x ejemplo y pegarlo acá quicktype.io y sino usar la extensión para vscode.

Entonces ahí ya nos va a tomar los datos si le definimos el tipo del body:   


Con axios:

Entre los <> se puede poner el tipo de dato de la response

A computer screen with text

Description automatically generated

Async await, lo que hace el async es convertir una función en una promesa. Estas dos funciones son exactamente iguales:

