# Curso de Programación funcional

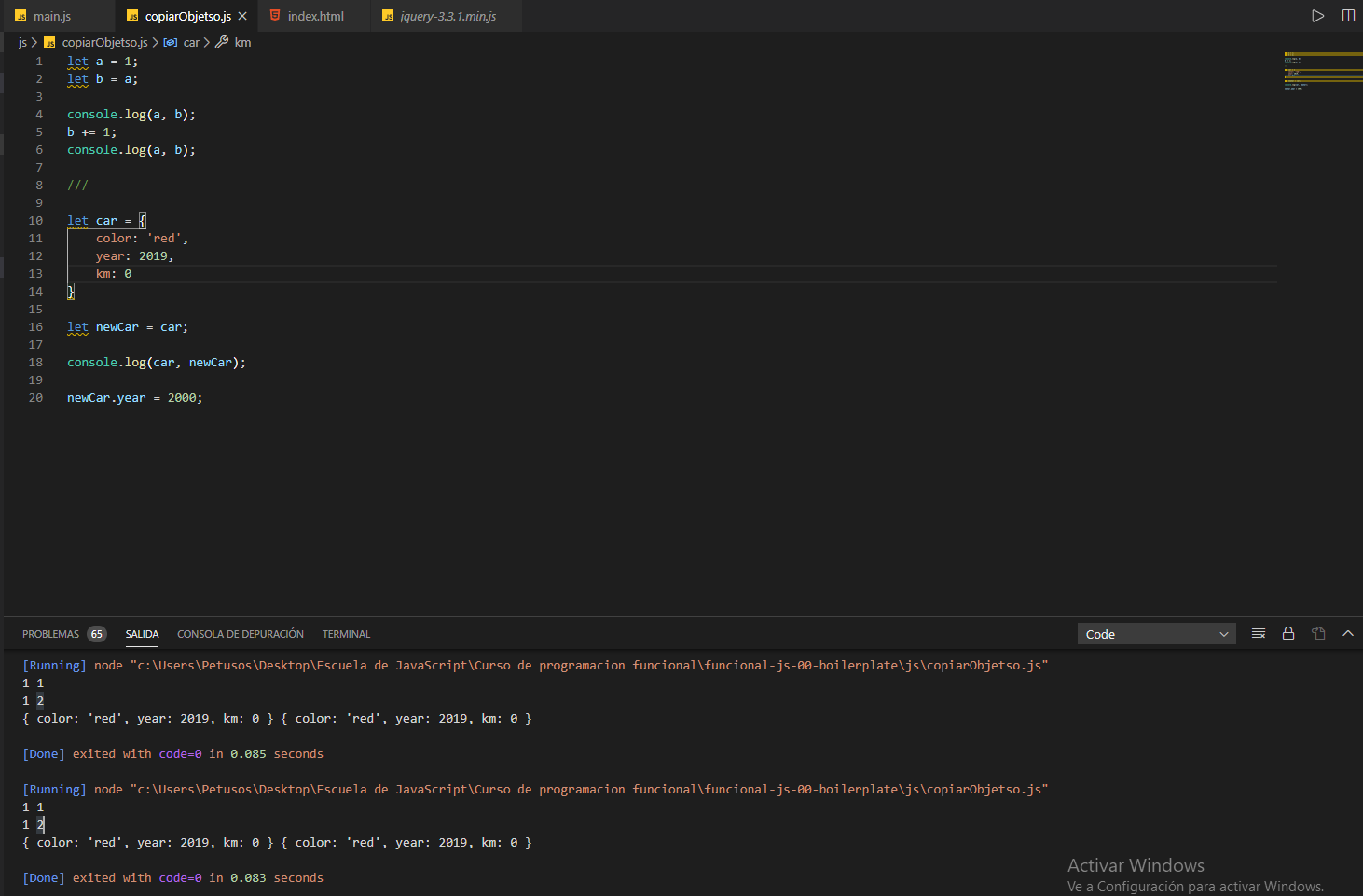
# Funciones puras

Conjunto de pasos finitos que trabajan únicamente con los parámetros de entrada que recibe y siempre regresa el mismo resultado para dichos parámetros de entrada. No interactúa con variables globales.

**Stack:** Donde se guardan los datos de tipo primitivo, cada vez que inicializamos o asignamos un tipo de variable primitiva. Es un tipo de memoria que usa LIFO.

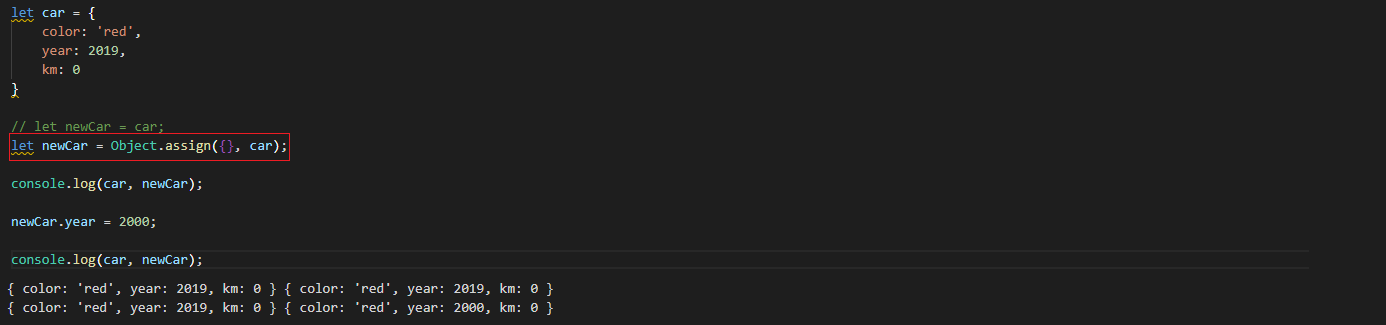
**Heap:** Más lento que el stack, son objetos. Es un espacio desordenado y no consecutivo al contrario que el stack. Cuando creamos algo en el HEAP también se crea en el stack un puntero a ese objeto del stack.

Vamos a ver este comportamiento con el siguiente código donde declaramos variables primitivas y luego una variable de tipo objeto.

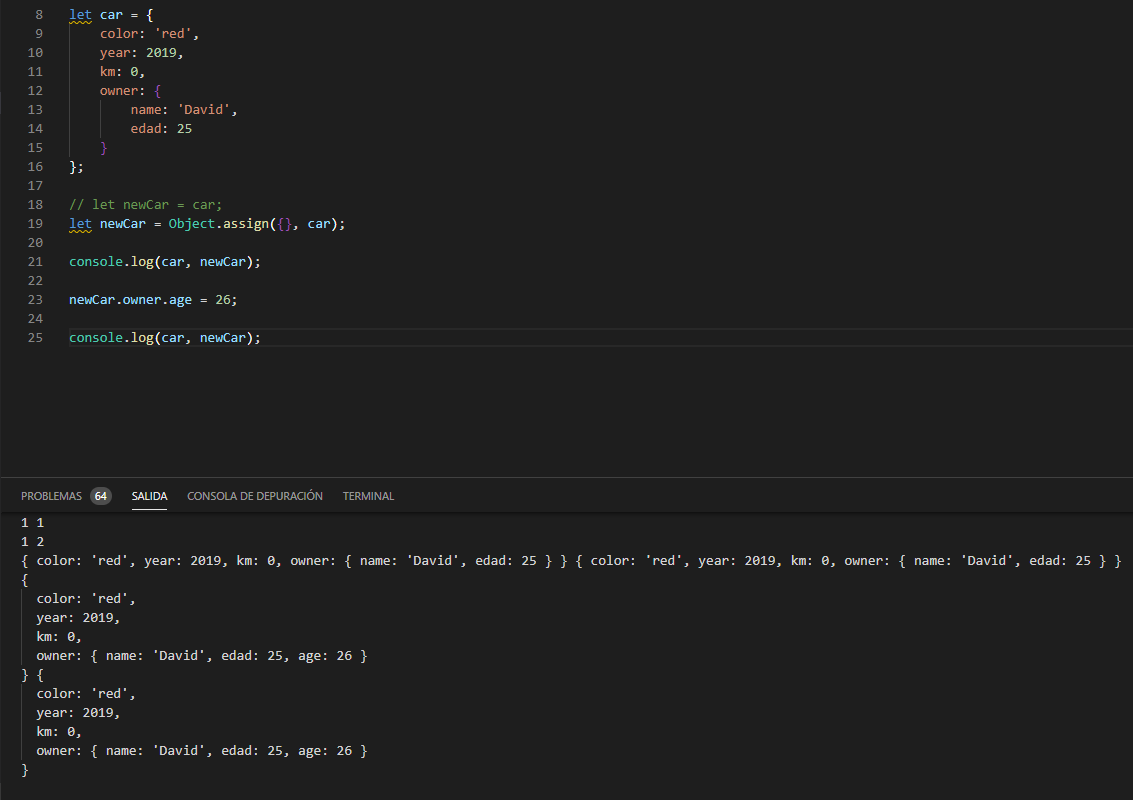


Como observamos cuando copiamos una variable del tipo primitiva simplemente estamos copiando los datos (stack) mientras que si copiamos el objeto estamos copiando la dirección de memoria del stack que apunta al HEAP así que si cambiamos algo de un objeto se cambia en el otro ya que apuntan al mismo.

Para corregir esto vamos a usar la funcion assign de Object que simplemente copia los valores de un objeto a otro

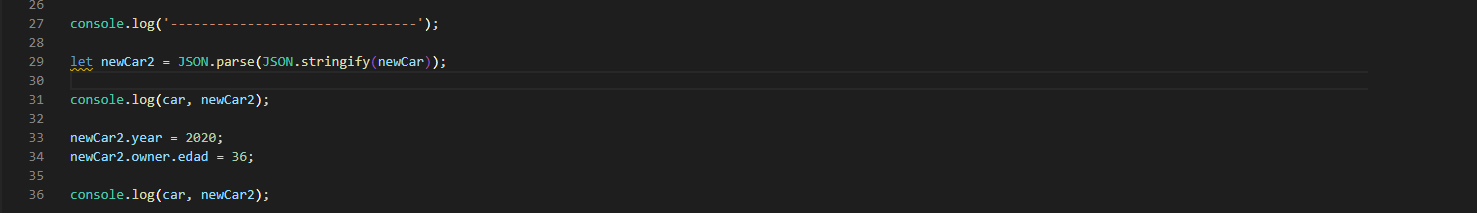


Pero object.Assign tiene un defecto no copia objetos dentro de otros más allá de un nivel.



Como puedes ver hemos cambiado la edad y se ha cambiado en los 2. Hay otras formas para poder hacer esto.

* **JSON.parse**: Convertir una cadena de texto en un objeto.
* **JSON.stringify()**: Convertir todos los objetos de javascript en un String.

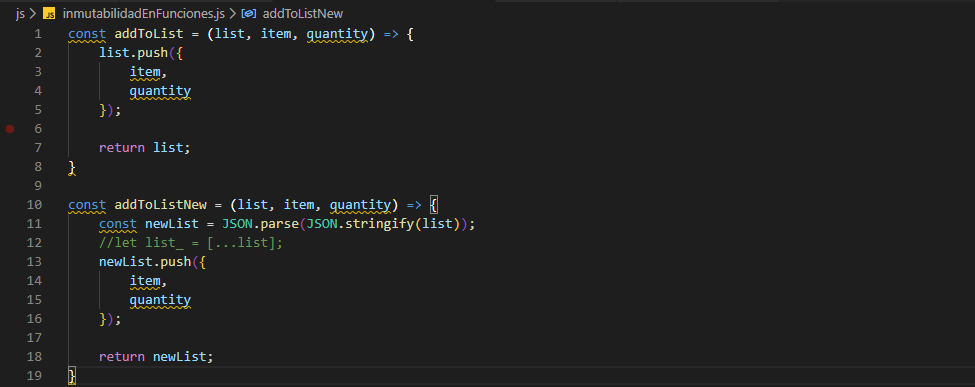


Básicamente estamos convirtiendo el objeto en una cadena, que se guarda en el stack y luego volviendo a convertirla en un objeto (nuevo) así simplemente al cambiar el objeto anidado como no apuntar al original, pues se cambia el dato.

# Inmutabilidad en nuestras funciones.

Normalmente en la programación funcional se intenta no modificar las variables que se nos pasan en las funciones, como puede ser un array que se le pasa como argumento.

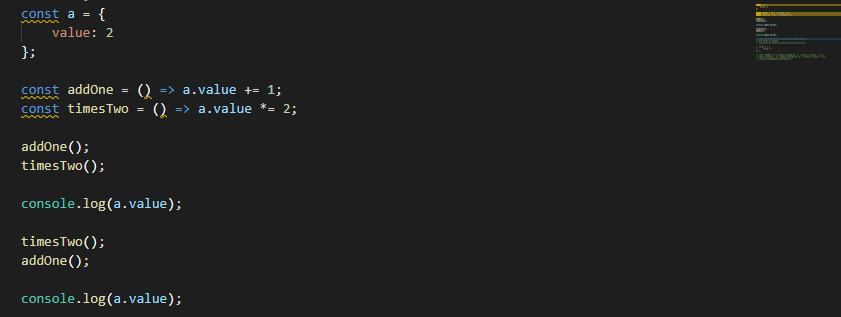
Véase el siguiente ejemplo.



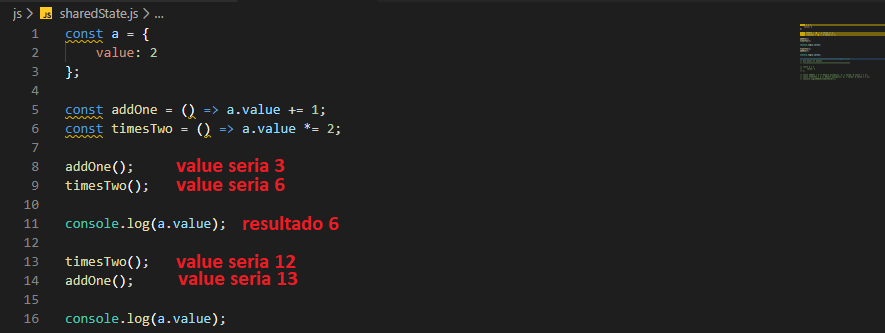
# Shared state

Una variable que tiene un scope compartido ya sea de manera local o global.

Vamos a verlo con un ejemplo.

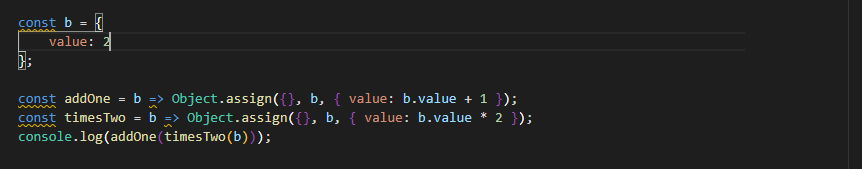


Cuando ejecutamos este código por consola tenemos el siguiente resultado :



Esto ocurre por que como vemos arriba, cada vez que llamamos a las funciones estamos pasándole la dirección del HEAP y por lo tanto modificamos el objeto original.

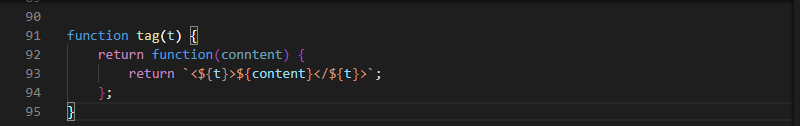
¿Cómo podemos hacer para no modificar el objeto original ‘a’?



Usamos Object.assign como ya aprendimos para copiar las propiedades de un objeto y modificar el objeto resultante.

# Aclaración de la clase 4.1 Funcion tag.

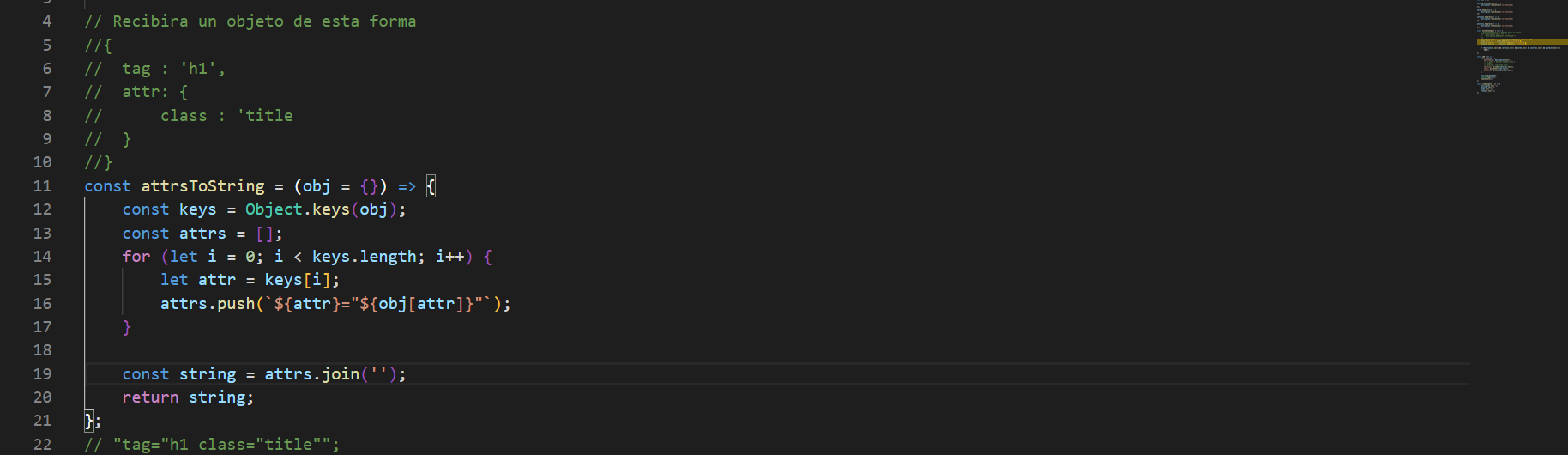
La funcion anterior es equivalente a lo siguiente:



La funcion anterior es un clousure.

# Funciones del proyecto.

## attrsToString



Esta es una funcion que recibe un objeto como se muestra arriba (comentado). Recorre las propiedades con un for y crea un arreglo de propiedades con el formato que tendría en HTML.

Es decir, si recibe como el código comentado seria tipo class=’title’

## tagAttrs



Esta funcion recibe un objeto y con ello forma la etiqueta completa.

# Currying