

```
> var Myvar="Hello"
< undefined
> console.log(Myvar)
Hello VM119:1
< undefined
> let Mylet = 60
< undefined
> console.log(Mylet)
60 VM191:1
< undefined
```

1)

```
> const Myconst = false
< undefined
> console.log(Myconst)
false VM338:1
< undefined
> Myvar = 100
< 100
> Mylet= "hi"
< 'hi'
> Myconst= true
✖ ▶ Uncaught TypeError: Assignment to constant variable.
  at <anonymous>:1:8 VM484:1
> |
```

It is possible to reassign and even alter the kinds of variables called var and let.

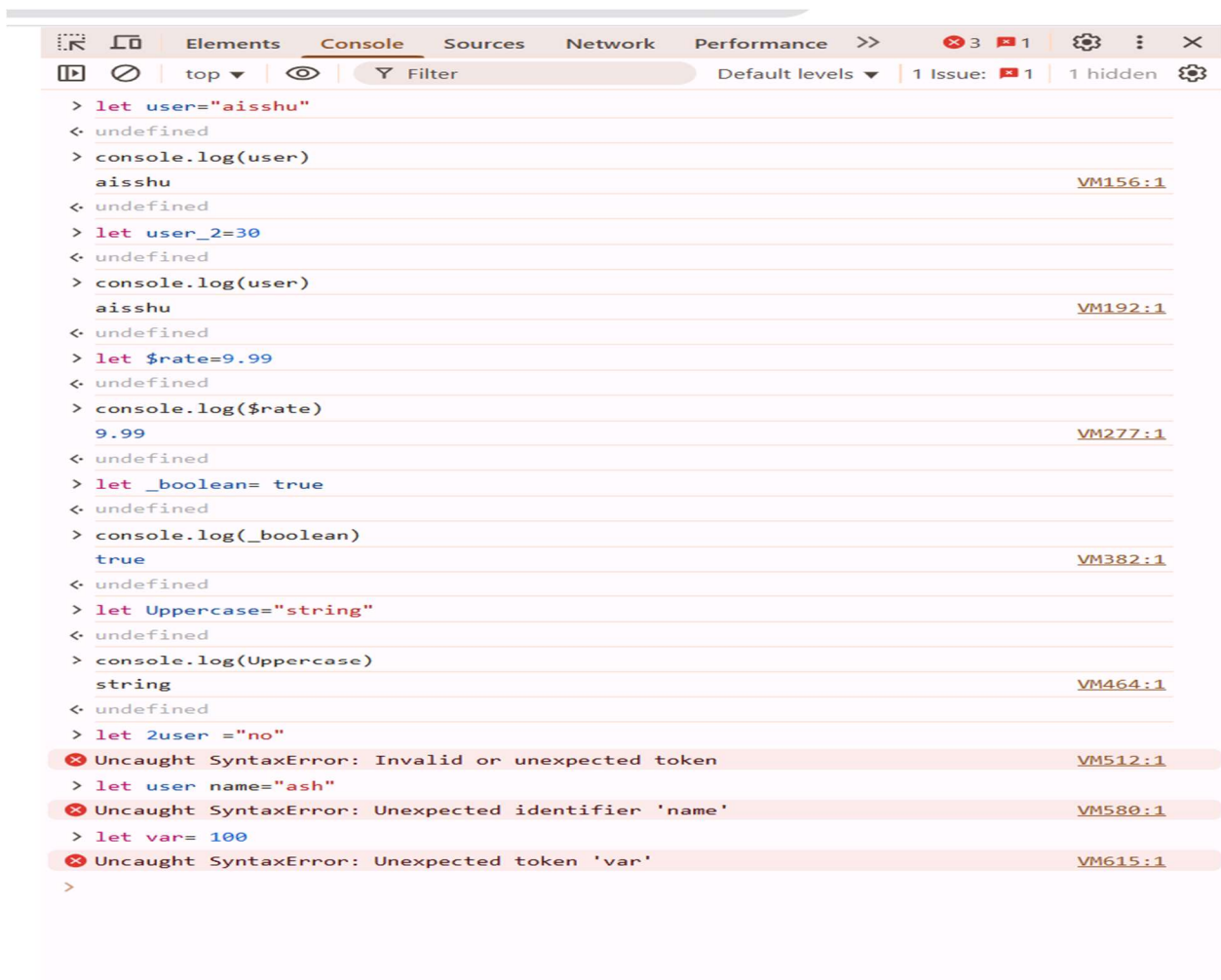
Once const is initialized, it cannot be reassigned. It must not change. Refer above screenshots

2) `let 2user = "no"` Variable names cannot begin with a number.

`let user name = "ash"` → Variable names cannot contain spaces.

`let var = 100` → `var` is not a valid variable name because it is a reserved keyword.

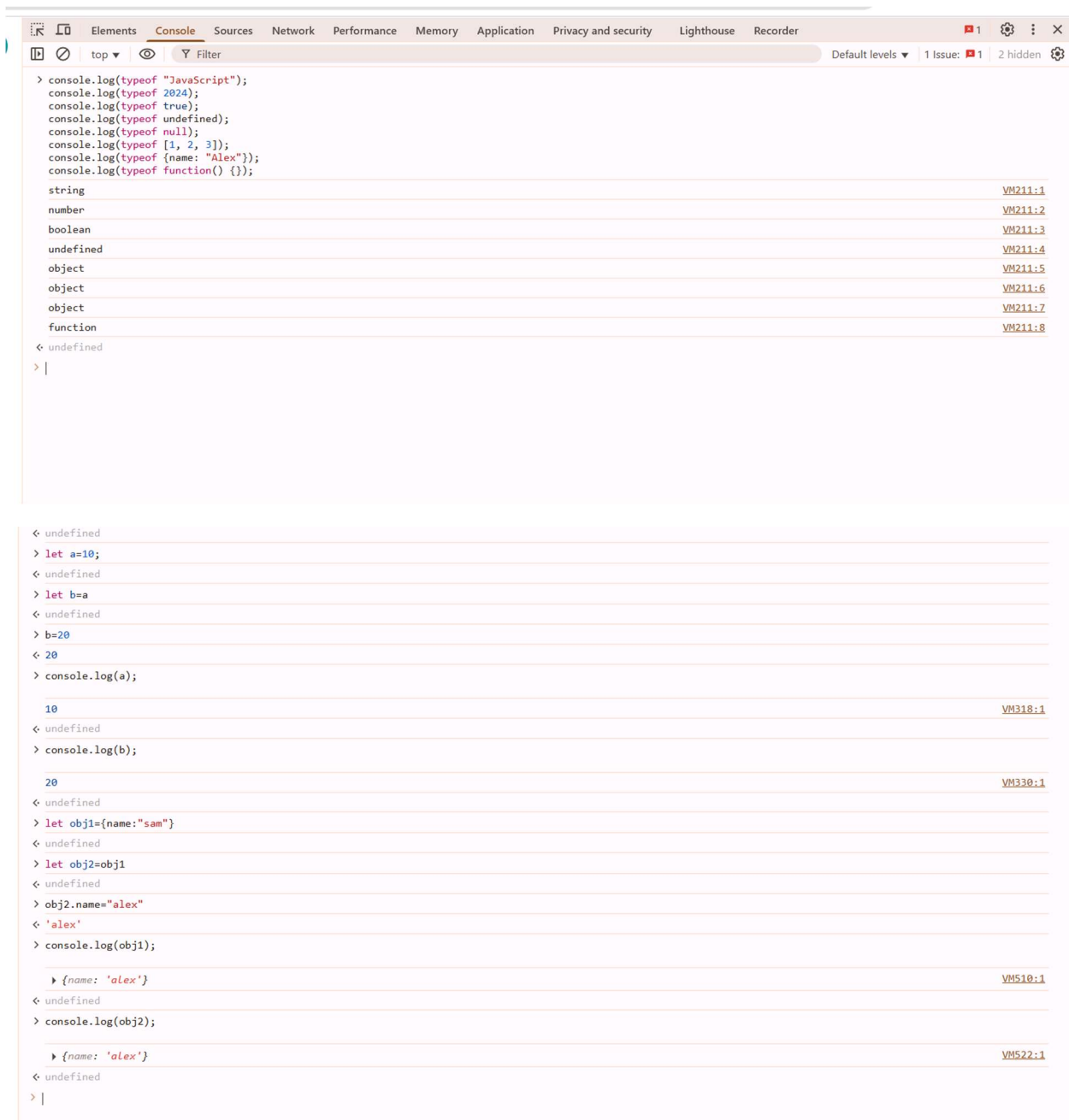
all others follow to correct JavaScript naming rules. Refer below screenshot



3) `typeof null` results in "object".

Arrays and functions are technically objects, but `typeof` gives "function" for functions.

Below screenshot



4) Primitive types are copied by value, therefore changing b has no direct effect on a. Obj1 and Obj2 both refer to the same memory location (i.e..copied by reference). Refer above screenshot

5) JavaScript is dynamically typed — variable types can change during runtime. Below screenshot

```
< undefined
> let Myvar=10
< undefined
> console.log(Myvar);

10 VM600:1
< undefined
> Myvar="hello"
< 'hello'
> console.log(Myvar);

hello VM641:1
< undefined
> Myvar={name:"alex"}
< {name: 'alex'}
> console.log(Myvar);

{name: 'alex'} VM702:1
< undefined
> |

< undefined
> let resultA = "5" + 3;
  console.log(resultA, typeof resultA);
  53 string VM706:2
< undefined
> let resultB = "5" - 2;
  console.log(resultB, typeof resultB);
  3 'number' VM710:2
< undefined
> let resultC = true + 1;
  console.log(resultC, typeof resultC);
  2 'number' VM714:2
< undefined
> let resultD = "hello" * 2;
  console.log(resultD, typeof resultD);
  NaN 'number' VM718:2
< undefined
> let resultE = null + 5;
  console.log(resultE, typeof resultE);
  5 'number' VM722:2
< undefined
> |
```

5- II)above screenshots

- a) Since 3 is forced into a string, "5" + "3" concatenates the strings. Thus, "53" type string(not addition) is the outcome.
- b) It forces the string "5" to become a number. Numerical subtraction is done by JavaScript: $5 - 2 = 3$.
- c) The boolean true is coerced to 1, so the expression becomes $1 + 1$, resulting in 2.
- d) JavaScript's attempt to translate "hello" to a number is unsuccessful. The result of "hello" * 2 is NaN (Not a Number), yet its type is still number because "hello" is not a numeric.
- e) null is coerced to 0 in numeric contexts. So, $0 + 5$ results in 5. Number type