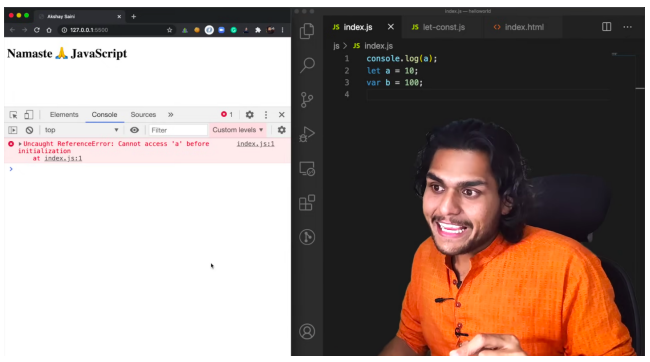
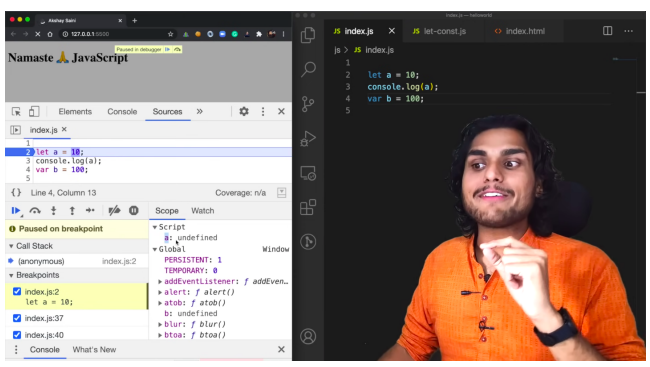
let and const in JS and Temporal Deadzone

Thursday, 17 August 2023 12:58 AM

→ let and const declarations are hoisted.



→ In above we can see that if we try to access 'a' before initialization then JS engine gives us reference error: cannot access a before initialization. But still let & const variables are hoisted. why?
This is because JS engine does allocate memory to let & const variables during memory creation phase but they are allocated memory in a different place than global. Thus, we can say that they are stored in a separate memory space.
So, we can't access let & const variables before their initialization.



(demo of above explanation)

Q: What is temporal deadzone?
ans. Temporal deadzone is the time since when let & const variables are hoisted till they are initialized some value.
Therefore, let & const variables are said to be in temporal deadzone till the time they are initialized with some value.
So, whenever we try to access a variable which is in temporal deadzone, it gives us a reference error.

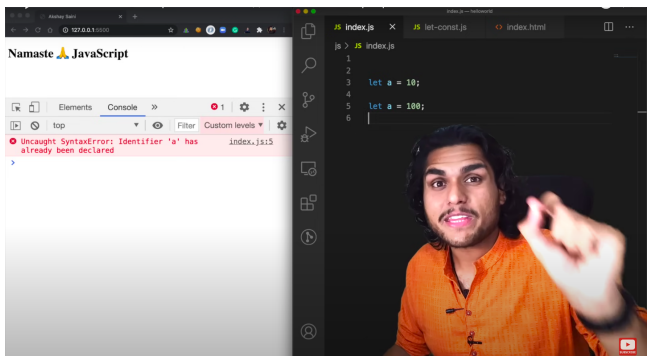
→ Reference error: We get reference error if JS engine does not find a particular variable inside the scope then it gives us: reference error: xc is not defined.

We also get reference error if we try to access variables which are in temporal deadzone before their initialization:
reference error: cannot access a before its initialization.

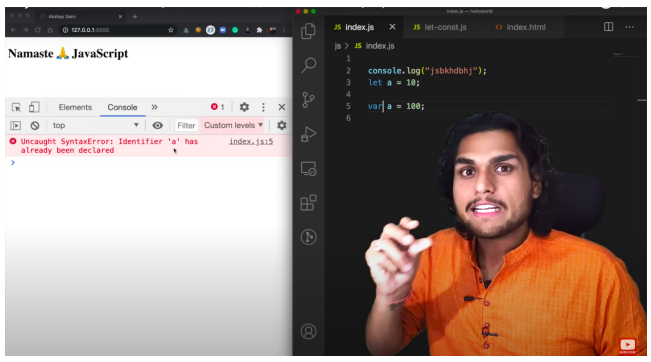
Note: let & const variables are not present on window object, so we cannot access them using window object as they are stored in a separate storage space.

→ let variables are stricter.

	var	let	Const
redeclaration	✓	✗	✗
reinitialization	✓	✓	✗
reassignment	✓	✓	✗

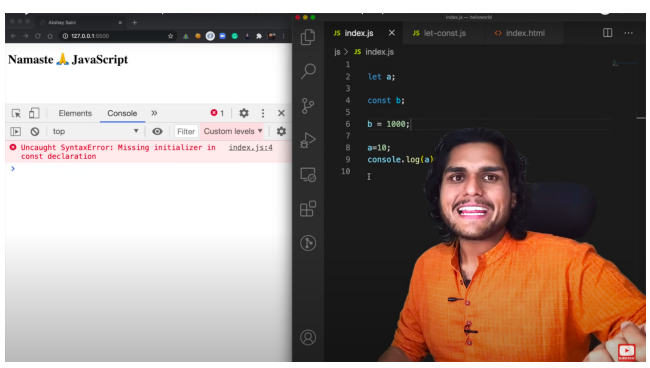


→ If we redeclare a let variable, JS engine gives us a syntax error.

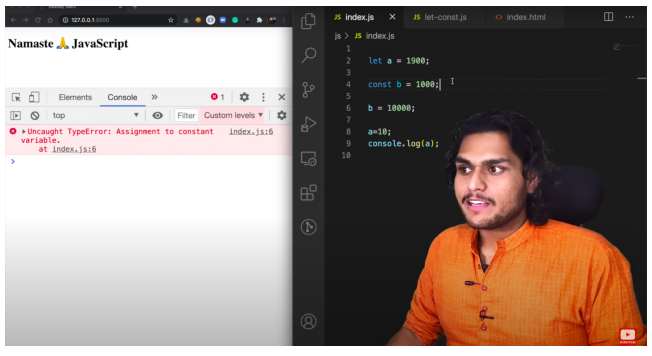


→ If we have declared a variable using let, then we can't redeclare it using var in the same scope.
→ However redeclaration is possible if we declared the variable using var in the first place.

Note: We can just declare a variable using let and initialize it later in the program. But this is not possible with const. const is even more stricter.



→ If we do not initialize a const variable then we will get a syntax error.



→ If we try to reassign a value to a const variable then JS engine gives us type error.

Note: Always try to use const and let. Avoid var declarations.

Q: How to avoid temporal deadzone?

ans. In order to avoid temporal deadzone, always keep all declarations and initializations on the top of the scope.