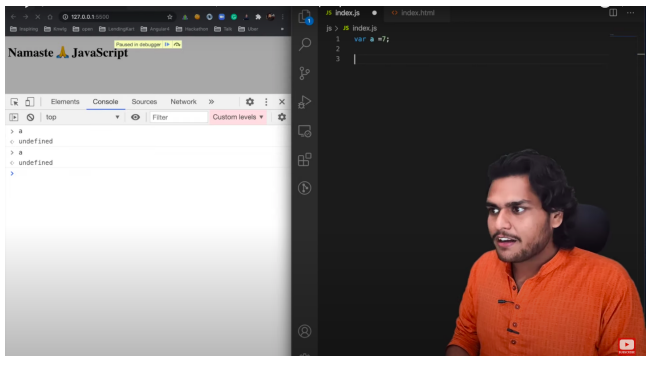
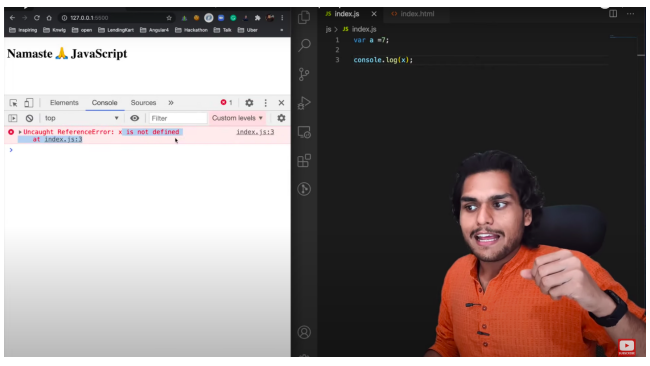
Undefined vs not defined in JS

Wednesday, 16 August 2023 10:50 PM

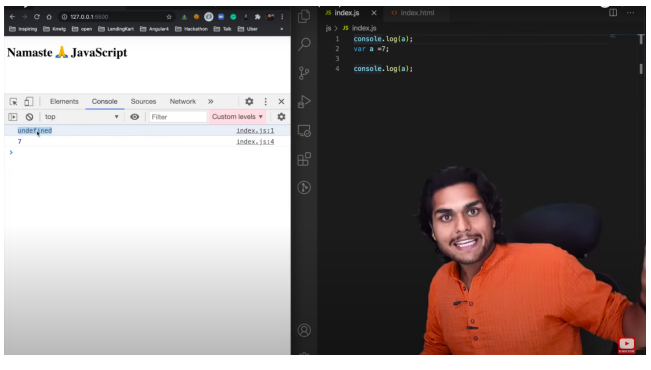
→ We have seen that JS tries to allocate memory to a variable in memory creation phase before the code is executed.
JS stores the variable with a special placeholder called undefined during the memory creation phase.



→ If we try to console 'x' then it gives us reference error; x is not defined this is because we have not yet allocated memory for x and yet we are trying to access it. So, rightnow x is not defined.



→ So the basic difference between undefined and not defined is that when we try to access a variable before its initialization then it gives us undefined, since JS has allocated memory for that variable in memory creation phase and given it a value of undefined.
whereas we get not defined when we try to access a variable which has not been allocated memory by JS engine yet.



→ In above code we get the following o/p, because in memory creation phase JS has allocated memory to 'a' and given it a value of undefined, so when we try to access the variable before its initialization then it gives us undefined. This is called hoisting.
(during code execution phase)

→ While we get o/p as '7' in line 4 because during code execution phase, value of 'a' has changed from undefined to 7 when JS engine runs line 2.

Note: Undefined does not mean empty. It is a special keyword which is given to a variable for the time being until it is initialised.

Note: JS is a loosely typed language. This means that JS does not attach itself to any particular data type. This means that if we put an integer in a variable, later on we can change its value and also put a string in it. Loosely typed languages are also known as weakly typed languages.

Note: we should not write like `var a = undefined;`
Although above code is not wrong but it is a bad practice to write like this, since undefined is a special keyword. This can lead to a lot of inconsistencies.