Class Notes 03-14-2020

Video 4  
Temporal Dead Zone – This exist because of scoping related to var variables. This goes back to rules all languages comply with and some languages like Pascal actually are built around. It also has to do with reference and value pointers. Which we won’t get into here but it is the reason this exist.

Video 5

There are different opinions on CONST and LET and VAR. The recommendation by the teacher and also me is this.

Assume you will always use CONST. Use LET when you know rebinding or meaning you will update the value. Avoid VAR at All times.

You will generally be able to spot people who have a background in languages like CLIPPER, VB 6, etc because they gravitate to VAR. This is what they know. But it is not a good idea at all. Also many examples on the web will be using VAR for the same reason. You should clean those up before you use them.

A CONST var is immutable. A LET var is changeable. However, if the CONST defines an object the properties are not immutable. Therefore, it is not necessary to avoid CONST for object declaration.

While ES6 does not include a way to make the entire object immutable there is a function from MDN called Freeze that will freeze all properties essentially making them CONST at a point in time. MDN grew out of what was originally the Mozilla Developer Network. Hence the name MDN.

IIFE is an Immediately-Invoked Function Expression - A JavaScript function that runs as soon as it is defined.

It is a design pattern which is also known as a Self-Executing Anonymous Function and contains two major parts:

* The first is the anonymous function with lexical scope enclosed within the Grouping Operator (). This prevents accessing variables within the IIFE idiom as well as polluting the global scope.
* The second part creates the immediately invoked function expression () through which the JavaScript engine will directly interpret the function.

Using LET and CONST are created at the BLOCK level so they are private to the code block they are part of by default scope.

You can avoid your VAR from leaking into the main stack by using let instead of var to define the variable.