



Episode 8 : let & const in JS, Temporal Dead Zone

- let and const declarations are hoisted. But its different from **var**

```
console.log(a); // ReferenceError: Cannot access 'a' before initialization
console.log(b); // prints undefined as expected
let a = 10;
console.log(a); // 10
var b = 15;
console.log(window.a); // undefined
console.log(window.b); // 15
```



It looks like let isn't hoisted, **but it is**, let's understand

- Both a and b are actually initialized as *undefined* in hoisting stage. But var **b** is inside the storage space of GLOBAL, and **a** is in a separate memory object called script, where it can be accessed only after assigning some value to it first ie. one can access 'a' only if it is assigned. Thus, it throws error.
- Temporal Dead Zone** : Time since when the let variable was hoisted until it is initialized some value.
 - So any line till before "let a = 10" is the TDZ for a
 - Since a is not accessible on global, its not accessible in *window/this* also. window.b or this.b -> 15; But window.a or this.a -> undefined, just like window.x->undefined (x isn't declared anywhere)
- Reference Error** are thrown when variables are in temporal dead zone.
- Syntax Error** doesn't even let us run single line of code.

o

```
let a = 10;
let a = 100; //this code is rejected upfront as SyntaxError. (duplicate
declaration)
-----
let a = 10;
var a = 100; // this code also rejected upfront as SyntaxError. (can't
use same name in same scope)
```

- **Let** is a stricter version of **var**. Now, **const** is even more stricter than **let**.

```
let a;  
a = 10;  
console.log(a) // 10. Note declaration and assigning of a is in different  
lines.  
-----  
const b;  
b = 10;  
console.log(b); // SyntaxError: Missing initializer in const declaration.  
(This type of declaration won't work with const. const b = 10 only will work)  
-----  
const b = 100;  
b = 1000; //this gives us TypeError: Assignment to constant variable.
```

- Types of **Error**: Syntax, Reference, and Type.
 - Uncaught ReferenceError: x is not defined at ...
 - This Error signifies that x has never been in the scope of the program. This literally means that x was never defined/declared and is being tried to be accessed.
 - Uncaught ReferenceError: cannot access 'a' before initialization
 - This Error signifies that 'a' cannot be accessed because it is declared as 'let' and since it is not assigned a value, it is its Temporal Dead Zone. Thus, this error occurs.
 - Uncaught SyntaxError: Identifier 'a' has already been declared
 - This Error signifies that we are redeclaring a variable that is 'let' declared. No execution will take place.
 - Uncaught SyntaxError: Missing initializer in const declaration
 - This Error signifies that we haven't initialized or assigned value to a const declaration.
 - Uncaught TypeError: Assignment to constant variable
 - This Error signifies that we are reassigning to a const variable.


SOME GOOD PRACTICES:

- Try using const wherever possible.

- If not, use let, Avoid var.
 - Declare and initialize all variables with let to the top to avoid errors to shrink temporal dead zone window to zero.
-

Watch Live On Youtube below:



 [Edit this page](#)