Hoisting In JavaScript

Hoisting simply gives higher specificity to javascript declarations. Thus, it makes the computer read and process declarations first before analyzing other code in program

Note → Hosting does not mean Javascript rearranges or move code above one another

Eg → `console.log(name) // Uncaught Reference Error`
`let name = 'Deepa';`

Variables declared with let and const are hoisted but not initialized with a default value.

Accessing let or const before it's declared will give :-

Uncaught Reference Error : cannot access before initialization

Remember the Error message tells variable is initialized somewhere

Variable hoisting with var

When interpreter hoists a variable declared with var, it initialized its value to undefined, unlike let or const.

Eg →
```
Console.log(name);   // undefined
var name = 'deepa';
console.log(name);   // 'deepa'
```

Now let's analyze this behaviour:

```
var name;
console.log(name);   // undefined
name = 'deepa';
console.log(name);   // deepa
```

Remember, the first console.log(name) outputs undefined becoz name is hoisted, and given a default value (not becoz variable is never declared).

Using undeclared variable will throw Reference Error

```
console.log(name);   // Uncaught ReferenceError: name is not defined
```

Date: / /

Now let's see if we donot declare var what happens

```
console.log(name);  // Uncaught ReferenceError
name = 'deepa';
```

↓

Assigning a variable that's not declared is valid

Javascript let us access variable before they're declared. This behaviour is an unusual part of javascript and can lead to errors.

Using variable before it's declaration is not desirable

The Temporal Dead Zone

The reason why we get reference Error when we try to access let or const before its declaration is

Temperal Dead Zone

Date: __/__/__

The TDZ starts at begining of the variables enclosing scope and ends when it is declared.

Accessing variable in TDZ gives Reference Error.

Eg
```
{  // start of foo's TDZ
    let bar = 'bar'
    console.log(bar); // 'bar'

    console.log(foo); // Reference Error
                      becoz we're in TDZ
    let foo = 'foo';
}  // End of foo's TDZ.
```

type of TDZ for let or const → Reference Error
for var ⇒ undefined

Functional Hoisting

Function declarations are hoisted too.
Function hoisting allows us to call function before it is declared or defined.

```
foo();                    // 'Foo'

Function foo() {
    console.log('foo');
}
```

Note only function declaration are hoisted not function Expressions.

Eg
```
foo();  // Uncaught Type Error:
var foo = function() {}
```

Uncaught Type Error: foo is not a function

```
bar(); // Uncaught Type Error
let bar = function() {}
```

Uncaught Reference Error: cannot access 'bar' before initialization

Similarly for const.

For function that is never declared:

```
foo(); // Uncaught Reference Error:
       foo is not defined
```