ES6 and Hoisting

**Hoisting** is the default behavior to move all of the declarations at the top of the scope before the execution of code. It applies to functions and variables. It allows the JavaScript to use the component before its declaration. Hoisting does not apply to scripts that run in strict mode.

ES6 and Variable Hoisting

Hoisting is the default behavior of JavaScript to move all declarations to the top of the current script, current function, or current scope. It allows you to use the variable before its declaration. JavaScript only hoists the variable declaration, not variable initialization.

**For example:**

**JavaScript Declarations are Hoisted**

1. x=10;
2. console.log(x);
3. var x;

Instead of giving a declaration error, the above code will successfully execute and show the desired output. It happens because of the hoisting concept. Let us see what happens when the code is in the compiling phase.

When the above code is in the compile phase, then it will be treated as:

**In Compile phase**

1. var x; // declaration of the variable will move on top.
2. x=10;
3. console.log(x);

**Output:**

10

JavaScript Initializations are not Hoisted

**1. When you initialize the variable before using it**

1. var x=100;
2. var y=200;
3. console.log(x+"  "+y);

**In compiling phase**

1. var x;
2. var y;
3. x=100;
4. y=200;
5. console.log(x+"  "+y);

**Output:**

100 200

**II. When you initialize the variable after using it**

1. var x=100;
2. console.log(x+"  "+y);
3. var y=200;

Let us see what happens when this code is in the compile phase.

When this code is in compiling, then it will be treated as follows:

**In Compiling phase**

1. var x;
2. var y;
3. x=100;
4. console.log(x+"  "+y);
5. y=200;

When you execute this code, you will get the following output in which the value of **y** is undefined.

**Output:**

100 undefined

This happens because hoisting does not allow us to move the initialization of variables on the top if you initialize them after using.