Hoisting is JavaScript's default behaviour of moving declarations to the top.

**JavaScript Declarations are Hoisted**

In JavaScript, a variable can be declared after it has been used.

In other words; a variable can be used before it has been declared.

**Example 1** gives the same result as **Example 2**:

Example 1

x = 5; // Assign 5 to x  
  
console.log(x)

var x; // Declare x

This declaration will execute as a first statement. So that console.log(x) print the value of x as 5

Example 2

var x; // Declare x  
x = 5; // Assign 5 to x

console.log(x)   // print the x in console

To understand this, you have to understand the term "hoisting".

Hoisting is JavaScript's default behaviour of moving all declarations to the top of the current scope (to the top of the current script or the current function).

The let and const Keywords-

Variables defined with let and const are hoisted to the top of the block, but not *initialized*.

Meaning: The block of code is aware of the variable, but it cannot be used until it has been declared.

Using a let variable before it is declared will result in a ReferenceError.

Example -This will result in a ReferenceError:

carName = "Volvo";  
let carName;

Using a const variable before it is declared, is a syntax error, so the code will simply not run.

Example-This code will not run.

carName = "Volvo";  
const carName;

**JavaScript Initializations are Not Hoisted**

JavaScript only hoists declarations, not initializations.

**Example 1** does **not** give the same result as **Example 2**:

Example 1

var x = 5; // Initialize x  
var y = 7; // Initialize y  
  
console.log(x + " " + y);           // Display x and y

Example 2

var x = 5; // Initialize x  
  
console.log(x + " " + y);

// Display x and y as undefined as it is not initializes.  
  
var y = 7; // Initialize y

Does it make sense that y is undefined in the above example?

This is because only the declaration (var y), not the initialization (=7) is hoisted to the top.

**Because of hoisting, y has been declared before it is used, but because initializations are not hoisted, the value of y is undefined.**

Example

var x = 5; // Initialize x  
var y;     // Declare y  
  
console.log(x + " " + y);

// Display x and y as undefined as the y is not initialized before you use y  
y = 7;    // Assign 7 to y

**Declare Your Variables At the Top !**