JavaScript Hoisting

Back to JS page

Table of Contents

- JavaScript Hoisting
  - JavaScript Declarations are Hoisted
    - Example 1
  - The let and const Keywords
    - Example 2
  - JavaScript Initializations are Not Hoisted
    - Example 3
  - Declare Your Variables At the Top
    - Example 4
  - Document
  - Reference

JavaScript Declarations are Hoisted

In JavaScript, a variable can be declared **after** it has been used. This is called **hoisting**.

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

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Hoisting</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>JavaScript Declarations are Hoisted</h4>
<p id="demo"></p>

<script>
// Using the variable BEFORE it is declared
// This works because var declarations are hoisted
x = 5;
document.getElementById("demo").innerHTML =
  "x is " + x + " (used before declaration!)";

var x; // Declare x - this is hoisted to the top
</script>

</body>
</html>
```

Example 1

Result View Example

The let and const Keywords

Variables defined with `let` and `const` are NOT hoisted like `var`. Using them before declaration causes an error.

This is because of the **Temporal Dead Zone** (TDZ) - variables declared with `let` / `const` are hoisted to the top of their block, but they are NOT initialized until their declaration statement is executed.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Hoisting</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>The let and const Keywords</h4>
<p id="demo1"></p>
<p id="demo2"></p>

<script>
// let and const are NOT hoisted like var
try {
  let carName = "Volvo";
  document.getElementById("demo1").innerHTML =
    "let carName: " + carName + " (declared before use - OK)";
} catch(err) {
  document.getElementById("demo1").innerHTML = "Error: " + err.message;
}

// Using let BEFORE declaration would cause an error
try {
  document.getElementById("demo2").innerHTML =
    "Attempting to use a let variable before declaration:";
  // This would throw: Cannot access 'x' before initialization
  console.log(city);
  let city = "New York";
} catch(err) {
  document.getElementById("demo2").innerHTML =
    "Error caught: " + err.message + "<br>" +
    "let/const are NOT hoisted like var";
}
</script>

</body>
</html>
```

Example 2

Result View Example

JavaScript Initializations are Not Hoisted

While **declarations** are hoisted, **initializations** (assignments) are not hoisted. The variable exists but has the value `undefined`:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Hoisting</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>JavaScript Initializations are Not Hoisted</h4>
<p id="demo"></p>

<script>
// The declaration of y is hoisted, but the assignment is not
// So y is undefined when we try to use it

function showHoisting() {
  document.getElementById("demo").innerHTML =
    "y is " + x + " and " + y + "<br><br>" +
    "x is 5 (declaration and initialization at top)<br>" +
    "y is undefined (only declaration hoisted, not value)";
}

var x = 5; // Declaration AND initialization
var y = 7; // Declaration is hoisted, initialization is not
showHoisting();
</script>

</body>
</html>
```

Example 3

Result View Example

Declare Your Variables At the Top

Because hoisting can cause unexpected results, it is good practice to always declare your variables at the top of the scope. This makes the code easier to read and understand:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Hoisting</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Declare Your Variables At the Top</h4>
<p id="demo"></p>

<script>
// Good practice: declare all variables at the top
var firstName = "John";
var lastName = "Doe";
var age = 30;
var city = "New York";

// Then use them below
document.getElementById("demo").innerHTML =
  "Good practice - declarations at top:<br><br>" +
  firstName + " " + lastName + "<br>" +
  "Age: " + age + "<br>" +
  "City: " + city + "<br><br>" +
  "This makes code easier to read and understand";
</script>

</body>
</html>
```

Example 4

Result View Example

Document

Document in project

You can Download PDF file.

Reference

- W3Schools JavaScript Hoisting