Hoisting In

@Samarthya Pandey

JS

Hoisting is a JavaScript mechanism that moves variable and function declarations to the top of their scope before code execution.

@Samarthya Pandey

This means that you can use a variable or a function before you declare it.

Before moving into Hoisting in more depth let's recap scopes in JavaScript.

@Samarthya Pandey

In JavaScript, there are two types of scopes: global and local. A global scope is the outermost scope that contains all other scopes. A local scope is created by a function or a block (with let or const).

Variables declared with var are hoisted to the top of their scope, whether it's global or local.

Variables declared with let or const are hoisted to the top of their block scope, but they are not initialized until their declaration is reached.

@Samarthya Pandey

This means that accessing them before their declaration will throw an error.

Functions in JavaScript can be declared in two ways: as function declarations or as function expressions.

@Samarthya Pandey

Function declarations are hoisted to the top of their scope and can be used before they are defined.

Function expressions are not hoisted and can only be used after they are assigned to a variable.

Here's an example of hoisting with var and function declaration:

@Samarthya Pandey

```
console.log(x);
// undefined
console.log(y);
// ReferenceError: y is not defined
console.log(add(2, 3));
// 5

var x = 10;
let y = 20;

function add(a, b) {
  return a + b;
}

// The above code is equivalent to:

var x;
function add(a, b) {
  return a + b;
}

console.log(x);
// undefined
console.log(y);
// ReferenceError: y is not defined
console.log(add(2, 3));
// 5

x = 10;
let y = 20;
```

JS

Here's an example of hoisting with let, const and function expression:

```
console.log(x);
// ReferenceError: x is not defined
console.log(y);
// ReferenceError: Cannot access 'y' before initialization
console.log(add(2, 3));
// TypeError: add is not a function

let x = 10;
const y = 20;

var add = function (a, b) {
  return a + b;
};

// The above code is equivalent to:

let x;
const y;
var add;

console.log(x);
// ReferenceError: x is not defined
console.log(y);
// ReferenceError: Cannot access 'y' before initialization
console.log(add(2, 3));
// TypeError: add is not a function

x = 10;
y = 20;

add = function (a, b) {
  return a + b;
};
```

@Samarthya Pandey

But remember Hoisting is not a real thing. It's just a way of describing how the JavaScript engine interprets your code.

@Samarthya Pandey

The engine scans your code for declarations (var, let, const and function) and creates them in memory before executing your code.

This is why you can use them before they appear in your code.

JS

Hoisting can make your code less readable and maintainable.

If you use variables or functions before they are declared, it can be hard to follow the logic and flow of your code. @Samarthya Pandey

It can also lead to unexpected results if you change the order or location of your declarations.

and

Also it can cause conflicts and bugs if you have multiple declarations with the same name in different scopes.