JavaScript Hoisting

Hoisting is a concept that enables us to extract values of variables and functions even before initializing/assigning value without getting errors and this happens during the 1st phase (memory creation phase) of the Execution Context.

Features of Hoisting:

- In JavaScript, Hoisting is the default behavior of moving all the declarations at the top of the scope before code execution. Basically, it gives us an advantage that no matter where functions and variables are declared, they are moved to the top of their scope regardless of whether their scope is global or local.
- It allows us to call functions before even writing them in our code.

Note: JavaScript only hoists declarations, not initializations.

JavaScript allocates memory for all variables and functions defined in the program before execution.

Sequence of variable declaration: The following is the sequence in which variable declaration and initialization occur.

Declaration –> Initialisation/Assignment –> Usage

Variable lifecycle:

```
let a;                  // Declaration
a = 100;              // Assignment
console.log(a);  // Usage
```

However, since JavaScript allows us to both declare and initialize our variables simultaneously, so we can declare and initialize at the same time.

```
let a = 100;
```

Note: Always remember that in the background the Javascript is first declaring the variable and then initializing them. It is also good to know that variable declarations are processed before any code is executed.

However, in javascript, undeclared variables do not exist until the code assigning them is executed. Therefore, assigning a value to an undeclared variable implicitly creates it as a global variable when the assignment is executed. This means that all undeclared variables are global variables.

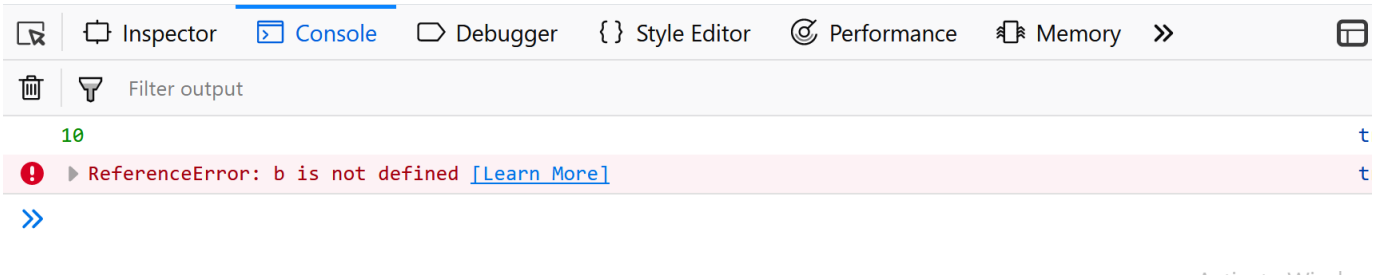
Example:

Javascript

```
// Hoisting
function codeHoist(){
    a = 10;
    let b = 50;
}
codeHoist();

console.log(a); // 10
console.log(b); // ReferenceError : b is not defined
```

Output:



Explanation: In the above code, we created a function called codeHoist() and in there we have a variable that we didn't declare using let/var/const and a let variable b. The undeclared variable is assigned the global scope by javascript hence we are able to print it outside the function, but in case of the variable b the scope is confined and it is not available outside and we get a ReferenceError.

Note: There's a difference between ReferenceError and undefined errors. An undefined error occurs when we have a variable that is either not defined or explicitly defined as type undefined. ReferenceError is thrown when trying to access a previously undeclared variable.

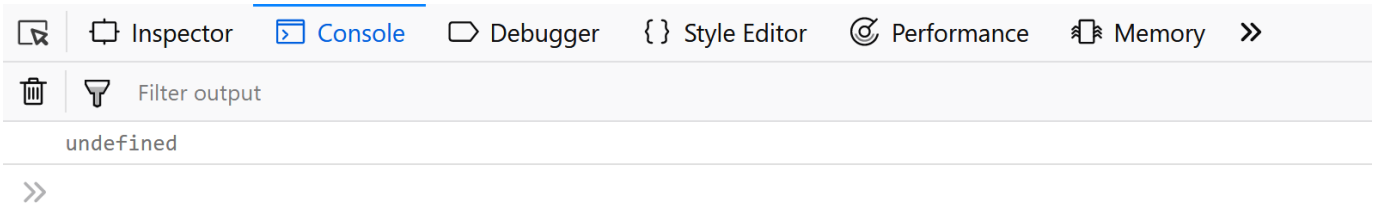
JavaScript var of ES5: When we talk about ES5, the variable that comes into our minds is var. Hoisting with var is somewhat different. When it is compared to let/const. Let's make use of var and see how hoisting works.

Example:

Javascript

```
// var code (global)
console.log(name); // undefined
let name = 'Mukul Latiyan';
```

Output:



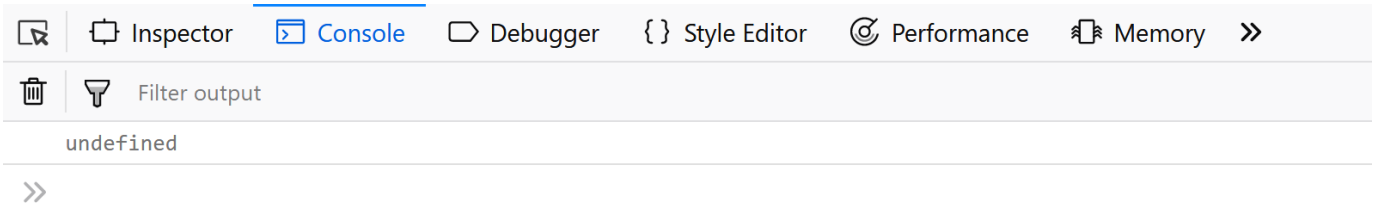
Explanation: In the above code we tried to console the variable name which was declared and assigned later, the compiler gives us *undefined* which we didn't expect as we should have gotten *ReferenceError* as we were trying to use the name variable even before declaring it.

But the interpreter sees this differently, the above code is seen like this:

Javascript

```
// how interpreter sees the above code
let name;
console.log(name); // undefined
name = 'Mukul Latiyan';
```

Output:



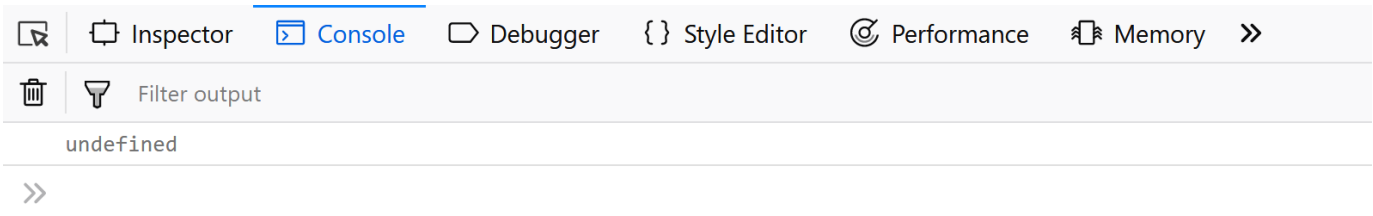
Function scoped variable: Let's look at how function-scoped variables are hoisted.

Example:

Javascript

```
// Function scoped
function fun(){
    console.log(name);
    let name = 'Mukul Latiyan';
}
fun(); // Undefined
```

Output:



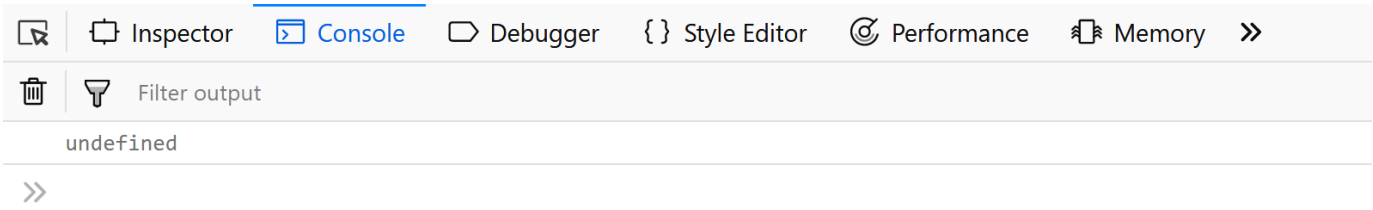
globally.

Example: We get undefined as the code seen by the interpreter.

Javascript

```
function fun(){
    let name;
    console.log(name);
    name = 'Mukul Latiyan';
}
fun(); // undefined
```

Output:



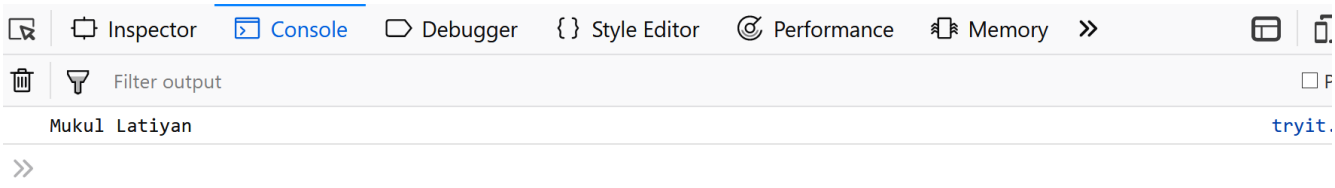
In order to avoid this pitfall, we can make sure to **declare and assign the variable at the same time, before using it.**

Example:

Javascript

```
// in order to avoid it
function fun(){
    let name = 'Mukul Latiyan';
    console.log(name); // Mukul Latiyan
}
fun();
```

Output:



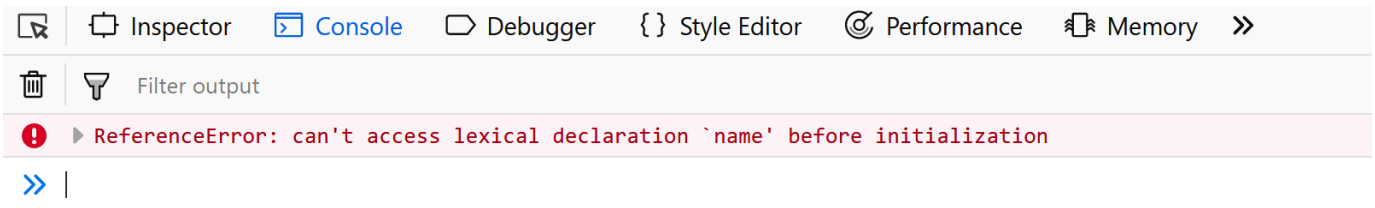
JavaScript Let of ES6: We know that variables declared with let keywords are block scoped not function scoped and hence there is no problem when it comes to hoisting.

Example:

Javascript

```
//let example(global)
console.log(name);
let name='Mukul Latiyan'; // ReferenceError: name is not defined
```

Output:



Explanation: Like before, for the var keyword, we expect the output of the log to be undefined. However, since the es6 let doesn't take kindly on us using undeclared variables, the interpreter explicitly spits out a Reference error. This ensures that we always **declare** our variable first.

JavaScript const of ES6: It behaves similarly to let when it comes to hoisting. A **function** as a whole can also be hoisted and we can call it before the declaration.

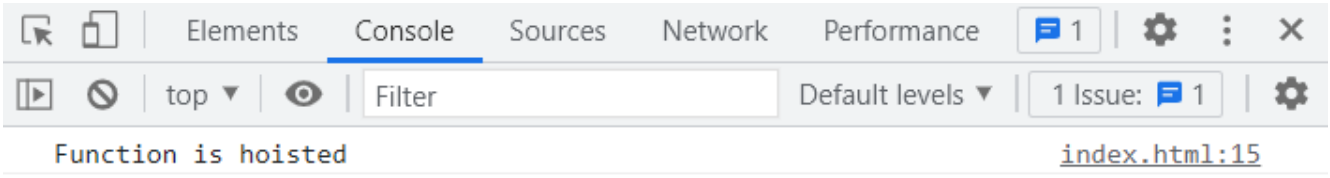
Example:

Javascript

```
fun(); // Calling before declaration

function fun(){ // Declaring
    console.log("Function is hoisted");
}
```

Output:



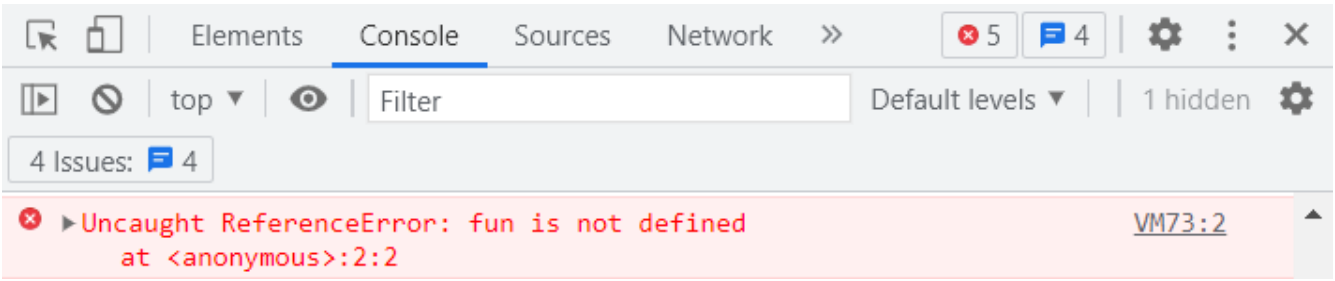
Also, if a function is used as an **expression** and we try to access it before the assignment an error will occur as only declarations are hoisted.

Example:

Javascript

```
fun() // Calling the expression

let fun = () =>{ // Declaring
    let name = 'Mukul Latiyan';
    console.log(name);
}
```



However, if var is used in the expression instead of let we will get the following Type Error as follows.

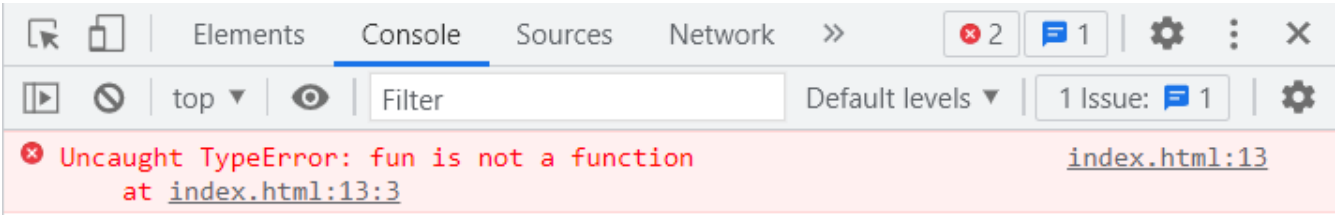
Example:

Javascript

```
fun() // Calling the expression

let fun = () =>{ // Declaring
    let name = 'Mukul Latiyan';
    console.log(name);
}
```

Output:



Last Updated : 22 May, 2023

78

Similar Reads

1. How does JavaScript Hoisting work internally ?
2. Scoping & Hoisting in JavaScript

4. JavaScript vs Python : Can Python Overtop JavaScript by 2020?
5. How to compare two JavaScript array objects using jQuery/JavaScript ?
6. How can JavaScript codes be hidden from old browsers that do not support JavaScript ?
7. How are the JavaScript window and JavaScript document different from one another?
8. Explore the concept of JavaScript Function Scope and different types of JavaScript Functions
9. Introduction to JavaScript Course - Learn how to build a task tracker using JavaScript
10. JavaScript Course What is JavaScript ?

Related Tutorials

1. Learn Data Structures with Javascript | DSA Tutorial
2. Onsen UI
3. React Material UI
4. NuxtJS
5. D3.js

Previous Next

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Improved By : Akanksha_Rai, immukul, shvrekhan, sweetyty, surinderdawra388, kbhattacharyya88, sumit_lovanshi, shobhit_sharma, amitstfcd

Article Tags : javascript-basics, JavaScript, Web Technologies

Improve Article | Report Issue

GeeksforGeeks
A-143, 9th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Android Tutorial

Algorithms

Sorting

Searching

Greedy

Dynamic Programming

Pattern Searching

Recursion

Backtracking

Web Development

HTML

CSS

JavaScript

Bootstrap

ReactJS

AngularJS

NodeJS

Computer Science

GATE CS Notes

Operating Systems

Computer Network

Database Management System

Software Engineering

Digital Logic Design

Engineering Maths

Python

Python Programming Examples

Django Tutorial

Python Projects

Python Tkinter

OpenCV Python Tutorial

Python Interview Question

Data Science & ML

Data Science With Python

Data Science For Beginner

Machine Learning Tutorial

Maths For Machine Learning

Pandas Tutorial

NumPy Tutorial

NLP Tutorial

Deep Learning Tutorial

DevOps

Git

AWS

Docker

Kubernetes

Azure

GCP

Competitive Programming

Top DSA for CP

Top 50 Tree Problems

System Design

What is System Design

Monolithic and Distributed SD

Top 50 String Problems

Top 50 DP Problems

Top 15 Websites for CP

High Level Design or HLD

Low Level Design or LLD

Top SD Interview Questions

Interview Corner

Company Preparation

Preparation for SDE

Company Interview Corner

Experienced Interview

Internship Interview

Competitive Programming

Aptitude

GfG School

CBSE Notes for Class 8

CBSE Notes for Class 9

CBSE Notes for Class 10

CBSE Notes for Class 11

CBSE Notes for Class 12

English Grammar

Commerce

Accountancy

Business Studies

Microeconomics

Macroeconomics

Statistics for Economics

Indian Economic Development

UPSC

Polity Notes

Geography Notes

History Notes

Science and Technology Notes

Economics Notes

Important Topics in Ethics

UPSC Previous Year Papers

SSC/ BANKING

SSC CGL Syllabus

SBI PO Syllabus

SBI Clerk Syllabus

IBPS PO Syllabus

IBPS Clerk Syllabus

Aptitude Questions

SSC CGL Practice Papers

Write & Earn

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships

Video Internship