Understanding Hoisting in JavaScript

Welcome to my presentation on Hoisting in JavaScript . In this session, we will explore Hoisting with Real Time Examples from Interview Point Of view .

by Gourav Roy

Content

- What is Hoisting
- Advantages Of Hoisting
- Variable Hoisting with Examples
- Function Hoisting with Examples
- Conclusion

What is Hoisting ?

Hoisting is a behavior in JavaScript where variable and function declarations are moved to the top of their containing scope during the compilation phase.

This means that regardless of where variables and functions are declared in the code, they are effectively "hoisted" to the top of their scope, making them available for use before their actual declaration in the code.

Advantages Of Hoisting

- Early Access to Functions
- Simplified Variable Declarations
- Enhanced Readability
- Consistent Behavior
- Supports Function Expressions

Variable Hoisting

```
console.log(x); // undefined
var x = 5;
console.log(x); // 5
```

The above code is effectively treated as if it were written like this during compilation:

```
var x;
console.log(x); // undefined
x = 5;
console.log(x); // 5
```

The declaration `var x;` is hoisted to the top, but the assignment (`x = 5;`) remains in place.

Function Hoisting

The function declaration is hoisted to the top, so the function can be called before its actual position in the code.

This is effectively treated as if it were written like this during compilation:

```
function sayHello() {
    console.log("Hello, world!");
}
sayHello(); // "Hello, world!"
```

It's important to note that hoisting does not occur with variables declared using `let` or `const`. They are hoisted, but unlike `var`, they are not initialized with `undefined` before the execution.

```
console.log(y); // ReferenceError
let y = 10;
```

In the case of functions declared using function expressions or arrow functions, they are not hoisted in the same way as function declarations. Function expressions are assigned at runtime, so they remain in place.

```
sayHi(); // TypeError: sayHi is not a function
var sayHi = function() {
    console.log("Hi!");
};
```

Conclusion

In conclusion, hoisting is a fundamental mechanism in JavaScript where variable and function declarations are moved to the top of their containing scope during the compilation phase. This behavior allows developers to use functions before their actual declarations in the code and provides a level of flexibility in organizing and structuring scripts.

A solid understanding of hoisting contributes to writing clean and predictable JavaScript code, promoting effective debugging and improved code comprehension.