Function Hoisting in JS:

In JavaScript, "hoisting" refers to the behavior where function and variable declarations are moved to the top of their scope before the code is executed. This means that a function or variable can be used before it is declared in the code. However, only the declaration is hoisted, not the assignment.

For example, the following code will work even though the function is called before it is declared:

```
example(); // Output: "This function has been hoisted."

function example() {
  console.log("This function has been hoisted.");
}
```

However, if you use a function expression instead of a function declaration, the function will not be hoisted, and the following code will throw an error:

```
example(); // ReferenceError: example is not defined

let example = function() {
  console.log("This function has not been hoisted.");
};
```

It's important to be aware of hoisting behavior when writing JavaScript code, as it can lead to unexpected results if not understood properly.

Variable Hoisting in JS:

In JavaScript, "hoisting" also applies to variable declarations. Variable declarations are moved to the top of their scope before the code is executed, meaning that a variable

can be used before it is declared in the code. However, like functions, only the declaration is hoisted, not the assignment.

For example, the following code will work even though the variable is used before it is declared:

```
console.log(example); // Output: undefined
var example = "This variable has been hoisted.";
```

Notice that the variable is undefined, because the variable declaration is hoisted to the top of the scope, but the assignment is still in the original position.

However, if you use let or const instead of var to declare variables, the variable will not be hoisted, and the following code will throw an error:

```
console.log(example); // ReferenceError: example is not defined
let example = "This variable has not been hoisted.";
```

It's important to be aware of hoisting behavior when writing JavaScript code, as it can lead to unexpected results if not understood properly.