Hoisting in JavaScript

Hoisting is the default behavior in JavaScript where declarations of variables and functions are moved to the top of their respective scopes during the compilation phase. This ensures that regardless of where variables and functions are declared within a scope, they are accessible throughout that scope.

Features of hoisting

- Declarations are hoisted, not initializations.
- Allows calling functions before their declarations.
- All variable and function declarations are processed before any code execution.
- Undeclared variables are implicitly created as global variables when assigned a value.

Hoisting var variables

Take a look at this example:

```
console.log(name)
// undefined

var name = "Dillion"
```

Here, we declare a variable called `name` with a string value of "Dillion". But, we try to access the variable before the line it was declared. But we don't get any errors. **Hoisting happened**. The `name` declaration is hoisted to the top, so the interpreter "knows" that there is a variable called `name`. If the interpreter did not know, you would get **name is not defined**. Let's try it out:

```
console.log(name)
```

```
// ReferenceError: name is not defined

var myName = "Dillion"
```

We have a variable called `myName` but no `name`. We get the **name is not defined** error when we try to access `name`. The interpreter "does not know" about this variable.

Coming back to our example above:

```
console.log(name)
// undefined

var name = "Dillion"
```

Although hoisting happened here, the value of `name` is undefined when we access it before the line of declaration. With variables declared `var`, the variable declaration is hoisted but with a default value of `undefined`. The actual value is initialized when the declaration line is executed.

By accessing the variable after that line, we get the actual value:

```
console.log(name)
// undefined

var name = "Dillion"

console.log(name)
// Dillion
```

Let's say we declared `name` in a function:

```
print()

console.log(name)
// ReferenceError: name is not defined

function print() {
  var name = "Dillion"
}
```

Here, we get a reference error: **name is not defined**. Remember, variables are hoisted but **only to the top of the scope they were declared in**. In this case, `name` is declared in `print`, so it will be hoisted to the top of that local scope. Let's try to access it in the function:

```
print()

function print() {
  console.log(name)
  // undefined

  var name = "Dillion"
}
```

By trying to access `name` in the function, even though it's above the line of declaration, we do not get an error. That's because `name` is hoisted, but don't forget, **with a default value of** `undefined`.