变量提升

Hoisting 是您在JavaScript文档中找不到的术语。Hoisting 被认为是思考执行上下文（特别是创建和执行阶段）在JavaScript中如何工作的一般方式。但是，hoisting 可能会导致误解。例如，提升教导变量和函数声明被物理移动到编码的顶部，但这根本不是什么。真正发生的什么是在编译阶段将变量和函数声明放入内存中，但仍然保留在编码中键入的位置。

> Note: Hoisting 真正发生的是在编译阶段将变量和函数声明放入内存中，但仍然保留在编码中键入的位置。

了解更多

技术范例

JavaScript 在执行任何代码段之前，将函数声明放入内存中的优点之一是，这允许你可以在你的代码中使用一个函数，在声明该函数之前。

例如：

```
/**
* 正确的方式：先声明函数，再调用函数 (最佳实践)
*/
function catName(name) {
    console.log("My cat's name is " + name);
}

catName("Tigger");

/*
The result of the code above is: "My cat's name is Tigger"
*/


/*变量提升*/

foo = 2;
var foo;

// 被隐式地解释为:

var foo;
foo = 2;
```

上面的代码片段是你希望编写代码以使其工作的方式。现在，我们来看看当我们在写这个函数之前调用这个函数会发生什么：

```
/**
* 不推荐的方式：先调用函数，再声明函数
*/

catName("Chloe");

function catName(name) {
    console.log("My cat's name is " + name);
}

/*
The result of the code above is: "My cat's name is Chloe"
*/

// 等价于

/*函数声明提升*/
function catName(name) {
    console.log("My cat's name is " + name);
}

catName("Tigger");

/*
The result of the code above is: "My cat's name is Chloe"
*/
```

即使我们先在代码中调用函数，在写该函数之前，代码仍然可以工作。这是因为在JavaScript中上下文如何执行的工作原理。

Hoisting 也适用于其他数据类型和变量。变量可以在声明之前进行初始化和使用。但是如果没有初始化，就不能使用它们。

技术范例

```
num = 6;
num + 7;
var num;
/* 没有给出错误，只要声明了num */
```

JavaScript 仅提升声明，而不是初始化。如果你使用的是在使用后声明和初始化的一个变量，那么该值将是 undefined。以下两个示例演示了相同的行为。

```
var x = 1;
// 声明 + 初始化 x

console.log(x + " " + y);
// y 是未定义的

var y = 2;
// 声明 + 初始化 y


//上面的代码和下面的代码是一样的



var x = 1;
// 声明 + 初始化 x

var y;
//声明 y

console.log(x + " " + y);
//y 是未定义的

y = 2;
// 初始化  y
```

技术参考

- JavaScript: Understanding the Weird Parts - Udemy.com Course
- var statement - MDN
- function statement - MDN